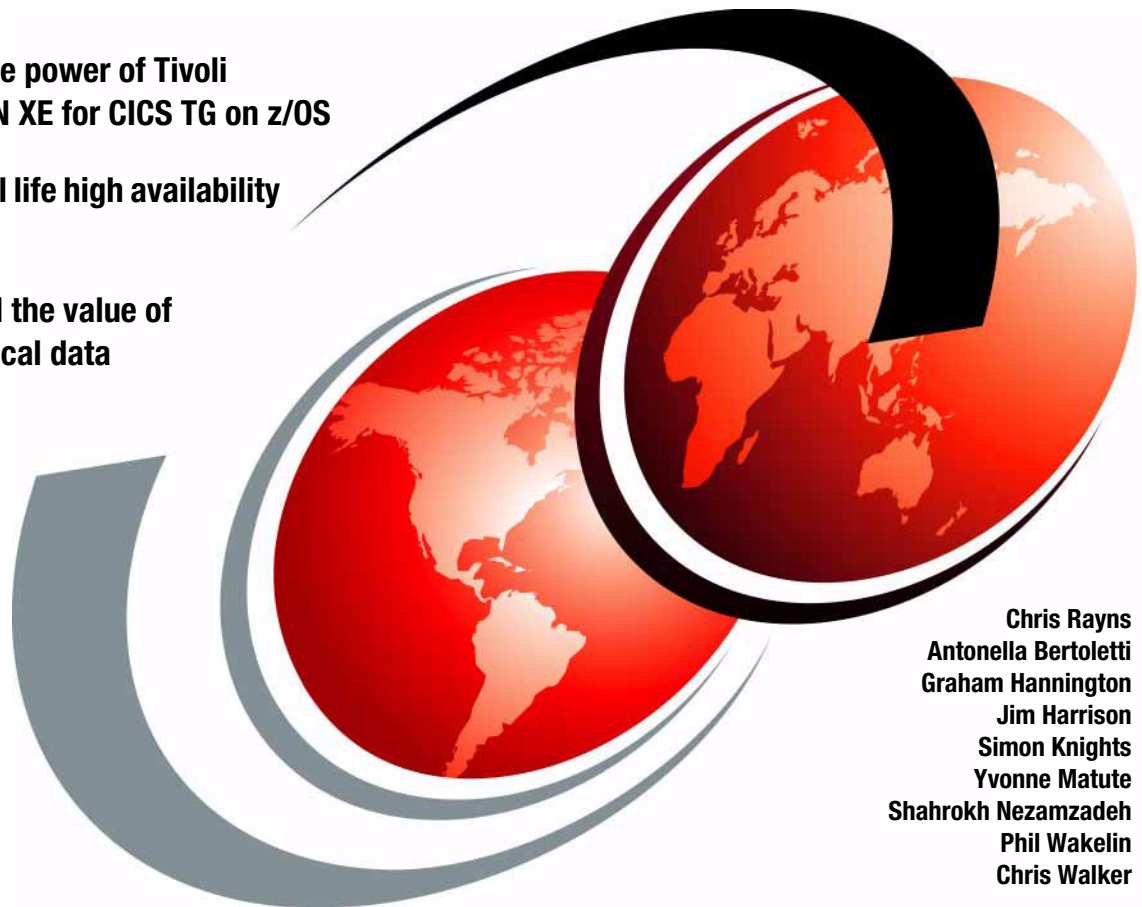


Exploring Systems Monitoring for CICS Transaction Gateway V7.1 on z/OS

Discover the power of Tivoli
OMEGAMON XE for CICS TG on z/OS

Explore real life high availability
scenarios

Understand the value of
SMF historical data



Chris Rayns
Antonella Bertoletti
Graham Hannington
Jim Harrison
Simon Knights
Yvonne Matute
Shahrokh Nezamzadeh
Phil Wakelin
Chris Walker



International Technical Support Organization

**Exploring Systems Monitoring for CICS Transaction
Gateway V7.1 on z/OS**

April 2008

Note: Before using this information and the product it supports, read the information in “Notices” on page ix.

First Edition (April 2008)

This edition applies to Version 7, Release 1, CICS Transaction Gateway.

© Copyright International Business Machines Corporation 2008. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	ix
Trademarks	x
Preface	xi
The team that wrote this book	xi
Become a published author	xiii
Comments welcome	xiii
Part 1. Introduction	1
Chapter 1. CICS Transaction Gateway V7.1	3
1.1 What CICS Transaction Gateway is	4
1.1.1 CICS TG products	4
1.1.2 CICS TG application programming interfaces	9
1.2 Local and remote modes of operation	12
1.2.1 Remote mode of operation	12
1.2.2 Local mode of operation	12
1.3 JCA	13
1.3.1 Overview of JCA	13
1.3.2 CICS resource adapters	16
1.4 Using the JCA with different CICS TG topologies	16
1.4.1 WebSphere Application Server and CICS TG: distributed platform ..	17
1.4.2 WebSphere Application Server on distributed, CICS TG on z/OS ..	19
1.4.3 WebSphere Application Server and CICS TG on System z	20
1.4.4 Mixed version support	22
Chapter 2. Configuring the CICS TG	25
2.1 Preparing for the installation	26
2.1.1 Software checklist	26
2.1.2 Basic test configuration	27
2.1.3 Definitions checklist	28
2.2 Installing CICS TG	28
2.2.1 Installing product files	28
2.2.2 Basic security considerations	30
2.3 Configuring CICS TG	33
2.3.1 Defining an HFS	33
2.3.2 Setting directory permissions	37
2.3.3 Creating a Gateway daemon configuration file	38
2.3.4 Creating an STDENV file	40

2.3.5	Creating the started task JCL	45
2.3.6	Configuring CICS TG for RRMS	46
2.3.7	Configuring CICS TG for SMF recording	49
2.4	Configuring CICS TS	52
2.4.1	Creating a CONNECTION definition	52
2.4.2	Creating a SESSIONS definition	53
2.4.3	Installing the definitions	54
2.4.4	Configuring CICS TS for RRMS	56
2.4.5	Compiling and installing the sample programs	56
2.4.6	Editing and assembling DFHCNV	56
2.4.7	Opening interregion communication	56
2.5	Testing the configuration	57
2.5.1	CTGTESTR sample program	57
2.5.2	Checking the Gateway daemon STDOUT	60
2.5.3	TCP/IP netstat command	60
2.5.4	Using the ping command	61
2.5.5	Running CTGTESTR	61
2.6	Operating CICS TG	62
2.6.1	Starting the Gateway daemon	62
2.6.2	Stopping the Gateway daemon	64
2.6.3	Configuring for multiple Gateway daemons	65
2.7	Migrating to CICS TG V7.1	69
2.7.1	Migrating from ctgenvar	69
2.7.2	Other migration and configuration considerations	70
2.8	Problem determination	71
2.8.1	Tips and utilities	71
2.8.2	Tracing	78
Chapter 3. Configuring OMEGAMON XE for CICS TG on z/OS		85
3.1	Preparing for the installation	86
3.1.1	Software checklist	86
3.1.2	The components of IBM Tivoli Monitoring	88
3.2	Installing OMEGAMON XE for CICS TG on z/OS	92
3.2.1	Installation considerations	92
3.2.2	Installing product files	92
3.2.3	Suggested publications	94
3.3	Configuring OMEGAMON XE for CICS TG on z/OS	95
3.3.1	Configuration considerations	95
3.3.2	Configuration on z/OS systems	97
3.3.3	Configuration on distributed systems	109
3.3.4	Historical configuration	112
3.3.5	Suggested publications	114

3.4 Testing the configuration	116
3.4.1 Verify the remote TEMS startup	116
3.4.2 Verify the OMEGAMON XE agent startup	117
Part 2. Problem determination scenarios	119
Chapter 4. Scenario environment	121
4.1 Introducing the environment	122
4.1.1 Software checklist	123
4.2 CICS TG and CICS configuration	124
4.2.1 CICS TG configuration	124
4.2.2 CICS configuration	126
4.3 OMEGAMON XE configuration	127
4.3.1 OMEGAMON XE components installed on z/OS	127
4.3.2 OMEGAMON XE components installed on Linux for System z	130
4.3.3 OMEGAMON XE components installed on Windows	134
4.3.4 Accessing data using the Tivoli Enterprise Portal client	140
4.4 WebSphere Application Server configuration	141
4.4.1 Resource Adapter creation	141
4.4.2 Connection factory creation	142
4.4.3 Connection factory customization	144
4.4.4 Application deployment	146
4.5 Workload and baseline performance	148
4.5.1 Workload used	148
4.5.2 Baseline performance figures	151
Chapter 5. Diagnosing common problems	153
5.1 Function used	154
5.2 Gateway daemon and ctgmaster errors	155
5.2.1 Gateway daemon fails to startup	156
5.2.2 Gateway daemon log errors at runtime	162
5.2.3 Ctgmaster fails to start up	163
5.3 Gateway and ECI return codes	165
5.3.1 ECI_ERR_NO_CICS -3	166
5.3.2 ECI_ERR_RESOURCE_SHORTAGE -16	169
5.3.3 ECI_ERR_SYSTEM_ERROR -9	175
5.3.4 ECI_ERR_SECURITY_ERROR -27	177
5.3.5 Invalid data returned in the COMMAREA	178
5.3.6 ERROR_WORK_WAS_REFUSED 61445 (0xF005)	178
5.4 Base Java API and J2EE exceptions	179
5.4.1 Java application hangs when trying to connect	179
5.4.2 IOException CTG6651E	186
5.4.3 javax.resource.ResourceException	186
5.4.4 javax.transaction.xa.XAException	188

5.4.5 IOException CTG6668E	188
Chapter 6. Diagnosing system slow downs	193
6.1 Scenario introduction.....	194
6.2 Functions used	196
6.2.1 OMEGAMON XE for CICS TG on z/OS	196
6.2.2 ctgping.....	197
6.2.3 Gateway daemon statistics	198
6.2.4 CICS TG monitoring exits	202
6.3 Problem scenarios.....	210
6.3.1 Network delay	211
6.3.2 Gateway daemon Worker thread constraint	213
6.3.3 Gateway daemon file system I/O constraint	222
6.3.4 CICS constraint.....	230
6.3.5 The delay only affects a minority of the requests	237
6.4 Summary.....	240
Chapter 7. High availability with XA and OMEGAMON XE	241
7.1 Scenario introduction.....	242
7.2 High availability: XA 1PC configuration.....	243
7.2.1 Using OMEGAMON XE	244
7.2.2 Gateway daemon unavailable.....	250
7.2.3 Gateway daemon failure	252
7.2.4 CICS region failure	260
7.2.5 Network failure	278
7.3 High availability - XA 2PC configuration	284
7.3.1 Using OMEGAMON XE	286
7.3.2 Gateway daemon unavailable.....	287
7.3.3 Gateway daemon failure	287
7.3.4 CICS region failure	289
7.3.5 Network failure	290
7.4 Summary.....	291
Chapter 8. Historical data analysis	293
8.1 Scenario introduction.....	294
8.2 Function used	294
8.2.1 OMEGAMON XE for CICS TG historical function	294
8.2.2 CICS TG SMF recording.....	295
8.3 Using historical data to diagnose system problems	297
8.3.1 Creating the base workload	298
8.3.2 Gathering historical data from SMF	301
8.3.3 Gathering historical data using OMEGAMON XE history data	302
8.3.4 Spotting the warning signs	304
8.3.5 Diagnosis using SMF statistics	305

8.3.6	Diagnosis using OMEGAMON XE	308
8.3.7	Action taken	324
8.4	Using CICS PA to analyze CICS TG SMF historical data	333
8.4.1	What CICS PA is	334
8.4.2	CICS PA support for analyzing CICS TG SMF 111 records	335
8.4.3	Viewing CICS TG statistics with CICS PA	337
8.4.4	Exporting CICS TG statistics to DB2	341
8.4.5	Exporting CICS TG statistics to spreadsheet applications	342
8.4.6	Charting CICS TG statistics	343
Part 3.	Appendixes	351
	Appendix A. Sample J2EE application - CTGTesterCCIXA	353
	The CTGTesterCCIXA application	354
	Application overview	354
	Appendix B. SMF Historical Data	365
	SMF interval statistics report	366
	Appendix C. Sample JCL for SMF reports	371
	JCL to link-edit CTGSMFRD	372
	JCL to extract type111 records from SMF	374
	JCL to run SORT SMF records	375
	JCL to run CICS TG record viewer sample program CTGSMFRD	375
	Appendix D. Additional material	379
	Locating the Web material	379
	Using the Web material	380
	System requirements for downloading the Web material	380
	How to use the Web material	381
	Related publications	383
	IBM Redbooks	383
	Other publications	383
	Online resources	384
	How to get Redbooks	384
	Help from IBM	384
	Index	385

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.


This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

Redbooks (logo) ®

eServer™

z/OS®

z/VM®

zSeries®

AIX®

Common User Access®

CICS/VSE®

CICS®

CICSplex®

CUA®

DB2®

DFSORT™

HiperSockets™

IBM®

IMS™

Language Environment®

MVS™

OMEGAMON II®

OMEGAMON®

Parallel Sysplex®

POWER™

Rational®

Redbooks®

RACF®

S/390®

SupportPac™

System z™

Tivoli®

TXSeries®

WebSphere®

The following terms are trademarks of other companies:

Oracle, JD Edwards, PeopleSoft, Siebel, and TopLink are registered trademarks of Oracle Corporation and/or its affiliates.

EJB, Java, Javadoc, JDBC, JDK, JNI, JSP, JVM, J2EE, Solaris, Sun, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Excel, Internet Explorer, Microsoft, SQL Server, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Itanium, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Preface

How can you manage your CICS® Transaction Gateway systems on z/OS® so as to meet the growing availability and performance requirements of your business?

The CICS Transaction Gateway V7.1 provides a wealth of new systems monitoring functionality, and together with the new IBM® Tivoli® OMEGAMON® XE for CICS TG product, provides you with a set of enhanced capacity planning and problem determination capabilities.

The first part of this IBM Redbooks® publication concentrates on product installation and customization for both the CICS Transaction Gateway V7.1 on z/OS and IBM Tivoli OMEGAMON XE for CICS TG product. The second part of the book highlights the new systems monitoring functionality in CICS Transaction Gateway V7.1 on z/OS and how it is complemented by the new IBM Tivoli OMEGAMON XE for CICS TG product. A set of typical customer scenarios are used to demonstrate the practical usage of the new functions, in the following four problem determination areas:

- ▶ Diagnosing common problems
- ▶ Diagnosing system slow downs
- ▶ High Availability with XA and OMEGAMON XE
- ▶ Historical data analysis

Together, these scenarios can be used to quickly determine the root cause of typical production problems and provide in-depth information to allow you to plan, build, and monitor a highly available CICS Transaction Gateway systems to provide SOA access to your existing CICS applications.

The team that wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Hursley Center.

Chris Rayns is the CICS Project Leader and writes extensively on all areas of CICS. He is an IT Specialist at the International Technical Support Organization, Poughkeepsie Center. Before joining the ITSO, Chris worked in IBM Global Services in the UK as a CICS IT Specialist.

Antonella Bertoletti is a Senior Certified Consulting I/T Specialist working for IBM Italy in Milan. She joined IBM in 1987. She specializes in the Business Process Management (BPM) and in the OLTP areas (TXSeries® and CICS, WebSphere® Application Server, and CICS Transaction Gateway). She has been involved in many different projects using OLTP products and distributed computing transactional technology, on several platforms, especially in the Insurance, Banking, and Manufacturing sectors. She is currently working in the IBM Software Group WebSphere Technical Sales Team, where she is the SWG BPM Integration Solution Architect for Italy. She holds a degree in Economics and Accounting from Pavia University.

Graham Hannington is a technical writer at Fundi Software, in Perth, Western Australia, where IBM CICS Performance Analyzer on z/OS (CICS PA) and IBM CICS Configuration Manager on z/OS (CICS CM) are developed. He has 19 years of experience in information development. He maintains the CICS PA user documentation, is the author of the CICS PA SupportPac™ CP12, *Charting historical CICS performance data*, and is the author of the *CICS CM User's Guide*. His areas of expertise include technical writing and illustration, XML-related technologies, such as XSLT, XML schema, and XML DOM programming, and VBScript and VBA programming.

Jim Harrison is a Software Support Specialist working for IBM United Kingdom, based in Warwick. He has worked in IBM since 1987, providing defect support for CICS products, including CICS TS, CICSplex/SM, and CICS tools. For seven years prior to joining IBM, he worked as a CICS/VSE® application programmer and CICS system programmer for an IBM customer in the retail sector. He holds a degree in Computer Science from Leicester Polytechnic and is a Fellow of the UKISA Technical Council.

Simon Knights is a Software Engineer in IBM UK in Hursley. He has 10 years of experience with the CICS Transaction Gateway. Since joining IBM in 1996, he has worked both as a developer and customer support specialist in the CICS Transaction Gateway group. He holds a degree in Physics from Bristol University.

Yvonne Matute is an Advisory Software Engineer in IBM USA. She has more than 20 years of experience in mainframe computing. She is currently the R&D team lead for the IBM Tivoli OMEGAMON XE for CICS and CICS TG products.

Shahrokh Nezamzadeh is an Advisory Software Engineer in IBM USA. He has more than 20 years of experience in mainframe computing. He is currently the technical lead for the Tivoli OMEGAMON XE for CICS TG product.

Phil Wakelin is the CICS Transaction Gateway Technical Architect, responsible for the product's future development. Based at IBM Hursley, he is an IBM Certified Solutions Expert in CICS Web Enablement and the author of many papers, IBM Redbooks, and CICS SupportPacs.

Chris Walker is a software engineer based at IBM Hursley. Since joining IBM in 2001, he has worked as both a system tester and developer for CICS Transaction Gateway and the Tivoli OMEGAMON XE for CICS and CICS TG products. He holds a Masters degree in Software Engineering from the University of Sheffield.

Thanks to the following people for their contributions to this project:

Steve Wall from the Montpelier PPSC for providing the CICDELAY sample.

Colin Alcock, Mark Ingamells, Richard Cowgill, Colin Westlake, Pete Masters and Tim Moran from IBM Hursley, Nigel Williams from IBM Montpelier and Silvia Ruksenas from IBM USA for review and technical assistance.

Rich Conway and Bob Hamowitz from the International Technical Support Organization, Poughkeepsie Center, for providing expert systems support.

Become a published author

Join us for a two- to six-week residency program! Help write a book dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Discover more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

ibm.com/redbooks

- ▶ Send your comments in an e-mail to:
redbooks@us.ibm.com
- ▶ Mail your comments to:
IBM Corporation, International Technical Support Organization
Dept. HYTD Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400



Part 1

Introduction

In this part, we give a broad overview of the CICS Transaction Gateway and Tivoli OMEGAMON XE for CICS TG on z/OS. We outline the major CICS TG components and describe the different topologies in which you can use the CICS TG. We also describe how we created our CICS TG and OMEGAMON XE environment for our scenario tests later in the book.



CICS Transaction Gateway V7.1

The CICS Transaction Gateway (CICS TG) is a market-leading connector that provides a high performing, secure, scalable, and tightly integrated method of e-business access to CICS. It enables CICS customers to expand the value of their investment in classic CICS applications and usually requires no changes to existing CICS applications. It is easy to install and has flexible configuration options that require minimal changes to CICS. It provides a range of networking options and provides a choice of Java™ and non-Java client programming interfaces.

In this chapter, we introduce the new function provided in the V7.1 release of the product, which is also discussed in the following chapters in this book.

1.1 What CICS Transaction Gateway is

The CICS TG is a set of client and server software components that allow a remote client application to invoke services in a CICS region. The client application can be either a Java application or a non-Java application using either C, C++, COBOL, or COM interfaces (depending on the platform used).

When a Java application is used, then the application can be any type of client (such as an applet, a servlet, or an enterprise bean). However, in the J2EE™ environment, the application is typically a servlet or enterprise bean that is deployed into a J2EE application server, such as WebSphere Application Server.

1.1.1 CICS TG products

With CICS TG V6 and later, there are now the following two orderable products:

- ▶ CICS TG on z/OS
- ▶ CICS TG for Multiplatforms

CICS TG on z/OS V7.1

CICS TG on z/OS V7.1 is the latest version of the z/OS product. It is supported on z/OS V1R6 and later and supports connectivity to CICS TS on z/OS V2.2 or later release (Figure 1-1).

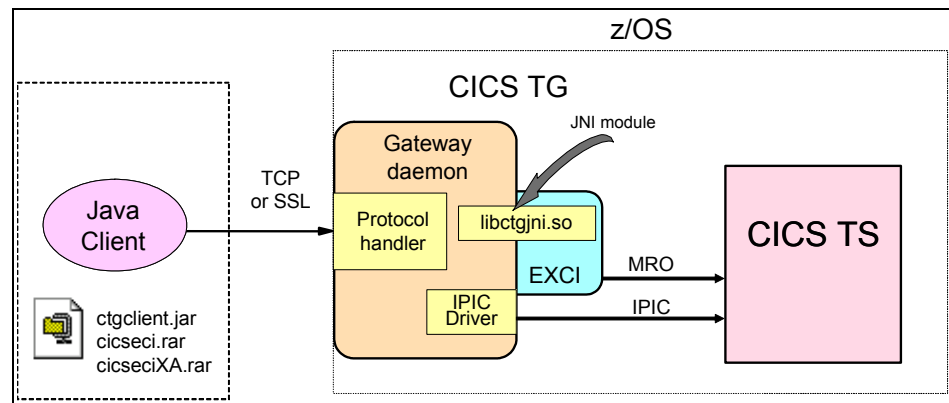


Figure 1-1 Components of CICS TG on z/OS

The product was announced on 6 November 2007 in a U.S. announcement letter, which is available at the following URL:

<http://www.ibm.com/common/ssi/cgi-bin/ssialias?infotype=an&subtype=ca&supplier=897&apname=IBMLinkRedirect&letternum=ENUS207-277>

CICS TG on z/OS uses either the external communication interface (EXCI) or the IP Interconnectivity Protocol (IPIC) to communicate with CICS regions on z/OS. It does not include the Client daemon and does not provide any support for non-Java based applications.

CICS TG on z/OS V7.1 has many improvements, principally in the areas of systems monitoring and extended integration with CICS TS.

- ▶ **Advanced system metrics:** Advanced capacity planning, throughput, and availability metrics are provided for the Gateway daemon. This function allows systems administrators and capacity planners to analyze system utilization metrics and to perform online problem determination. Over 100 statistics are now available, providing information about network usage statistics for EXCI and IPIC connections to CICS, usage of critical system resources, including region storage and Java Virtual Machine (JVM™) heap, and analysis of response times and network throughput. If necessary, these values can be used to take action to reduce the need for planned outages or prevent the occurrence of unplanned downtime. These statistics are made available through the extended z/OS system command-based administration interface and the external statistical API. This function is utilized in Chapter 6, “Diagnosing system slow downs” on page 193 and in Chapter 7, “High availability with XA and OMEGAMON XE” on page 241.
- ▶ **Interval statistics and off-line recording:** A mechanism is provided to record and reset statistics using a configurable interval. Statistics are recorded to the z/OS System Monitoring Facility (SMF) using a new SMF type 111 record. This allows analysis of trends and peak usage, enhancing the ability to perform capacity planning and performance monitoring of the Gateway daemon and providing for integration with the statistical collection policies and procedures used for the monitoring of CICS TS (see Chapter 6, “Diagnosing system slow downs” on page 193 for more information).

Note: Support for the analysis of the CICS TG SMF 111 records is also provided by CICS PA V2.1 (the service channel by PTF for APAR PK53163). This provides online dialog reporting and long-term historical collection for capacity planning and trend analysis purposes using the CICS PA Historical Database (HDB) facility.

- ▶ **Transaction monitoring:** A new request monitoring exit infrastructure is provided in the CICS TG for use in both local and remote Gateway scenarios. This infrastructure enables customers and Independent Software Vendors (ISVs) to develop transaction monitoring solutions for online transaction tracking and offline auditing. The exit infrastructure is provided in both the Java client and the Gateway daemon, and can be used to report response times and additional key information for tracking of ECI-based requests as

they flow through the CICS TG components. This function is also detailed in Chapter 6, “Diagnosing system slow downs” on page 193.

- ▶ **Advanced workload monitoring:** As part of the exploitation of CICS TS V3.2 IPIC protocol, support is provided for EWLM over IPIC connections to CICS TS V3.2. EWLM is the IBM implementation of the Application Response Measurement (ARM) standard from The Open Group. EWLM extends the capabilities of z/OS Workload Management (WLM) services to all members of the IBM eServer™ family, making possible end-to-end workload monitoring in heterogeneous environments, such as a WebSphere Application Server and CICS TS environment. Additionally, IPIC requests originating in a Java client will automatically contain CICS point-of-origin information, enabling CICSplex® SM (or equivalent CICS monitoring tools) to perform problem determination and offline analysis of requests as they enter and flow across a CICSplex.
- ▶ **Channels as modern-day COMMAREAs:** Interoperation with the CICS TS V3 channels and containers programming model provides an improved method of exchanging data with CICS programs, in amounts that far exceed the 32 KB limit that applies to COMMAREAs and additionally provides an optimized and more structured data interface. Support is provided for both the JCA ECI resource adapter and the CICS TG base classes and is conditional upon the use of the IP Interconnectivity (IPIC) protocol provided in CICS TS V3.2.
- ▶ **Extended XA support:** Deployment options when using XA support with the CICS ECI resource adapter are expanded to support both IPIC and EXCI connections when using the CICS TG on z/OS. XA transaction support enables CICS Transaction Server (CICS TS) on z/OS to participate in a global two-phase commit transaction that is initiated in a distributed J2EE V1.4 application server, such as WebSphere Application Server V6.0.
- ▶ **Extended Secure Sockets Layer (SSL) support:** Options for secure intercommunication are enhanced through the exploitation of the CICS TS V3.2 support for SSL/TLS, when using a local mode CICS TG on any supported platform. This feature enables Java clients to use an encrypted connection to a CICS TS V3.2 system, providing for secure transmission of data and optionally for authentication using X509 certificates.
- ▶ **WLM health reporting interval:** A configurable interval for reporting of health to the z/OS workload manager (WLM), allowing for faster recovery in TCP/IP load balancing scenarios. For more details, see Chapter 7, “High availability with XA and OMEGAMON XE” on page 241.
- ▶ **Request level timeout:** ECI requests using IPIC connections to CICS TS V3.2 can now specify a timeout value per request, allowing for fine grained control at the individual request level.

CICS TG for Multiplatforms V7.1

CICS TG for Multiplatforms is supported on the following range of operating systems and platforms and is designed to support connectivity to all in-service CICS servers:

- ▶ Linux® on system Z
- ▶ Linux on Intel®
- ▶ Linux on POWER™
- ▶ AIX®
- ▶ HP-UX (on PA-RISC and Itanium®)
- ▶ Sun™ Solaris™ (on SPARC)
- ▶ Windows® XP, Windows 2000, and Windows 2003 and Windows

This version was announced on 6 November 2007 in a U.S. announcement letter, which is available at:

<http://www.ibm.com/common/ssi/cgi-bin/ssialias?infotype=an&subtype=ca&supplier=897&appname=IBMLinkRedirect&letternum=ENUS207-274>

CICS TG for Multiplatforms is comprised of the following main runtime components:

- ▶ The Gateway daemon, which listens for incoming work and manages the threads and connections necessary to ensure good performance.
- ▶ The Client daemon, which provides the communication to CICS servers and the non-Java APIs.
- ▶ A Java class library or JCA resource adapter, which is deployed into the client runtime environment. When used in a JCA environment, the resource adapter is deployed into the J2EE application server.

A Java client program can connect to a remote Gateway daemon using the TCP or SSL protocols. The Client daemon then provides the transport drivers to connect to the CICS server, as shown in Figure 1-2 on page 8. Note that because the non-Java APIs are provided by the Client daemon, there is no remote connectivity support for non-Java clients.

Note: The CICS Universal Client V7.1, which provides the Client daemon functions within CICS TG, is available as a separate product on the AIX, Linux, and Windows platforms. CICS Universal Client provides the transport protocols and non-Java programming interfaces for single user access to CICS.

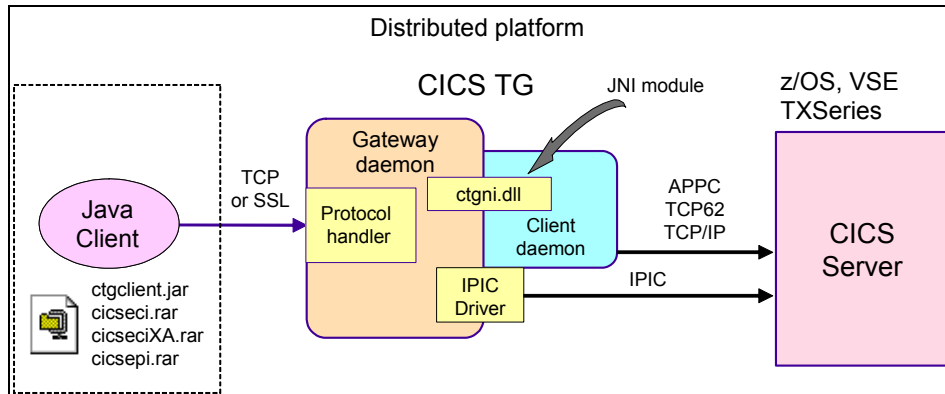


Figure 1-2 Components of CICS TG for Multiplatforms

CICS TG for Multiplatforms V7.1 provides the following improvements:

- ▶ **Advanced system metrics:** Advanced capacity planning, throughput, and availability metrics are provided for the Gateway daemon on all platforms. This function allows systems administrators and capacity planners to analyze system utilization metrics and to perform online problem determination. Over 100 statistics are now available, providing information about network usage statistics for SNA, ECI/IP, and IPIC connections to CICS, usage of critical system resources such as Java Virtual Machine (JVM) heap, and analysis of response times and network throughput. If necessary, these values can be used to take action to reduce the need for planned outages or prevent the occurrence of unplanned downtime. These statistics are made available through the extended *ctgadmin* command-based administration interface and the external statistical API.
- ▶ **Interval statistics and off-line recording:** A mechanism is provided to record and reset statistics using a configurable interval. This allows analysis of trends and peak usage, enhancing the ability to perform capacity planning and performance monitoring of the Gateway daemon and providing for integration with the statistical collection policies and procedures used for the monitoring of CICS TS. Note that interval statistics are not provided for the statistics provided for the Client daemon component.
- ▶ **Transaction monitoring:** A new request monitoring exit infrastructure is provided in the CICS TG for use in both local and remote Gateway scenarios. This infrastructure enables customers and Independent Software Vendors (ISVs) to develop transaction monitoring solutions for online transaction tracking and offline auditing. The exit infrastructure is provided in both the Java client and the Gateway daemon, and can be used to report response times and additional key information for tracking of ECI-based requests as they flow through the CICS TG components.

- ▶ **Advanced workload monitoring:** As part of the exploitation of the CICS TS V3.2 IPIC protocol, support is provided for EWLM over IPIC connections to CICS TS V3.2. EWLM is the IBM implementation of the Application Response Measurement (ARM) standard from The Open Group. EWLM extends the capabilities of z/OS Workload Management (WLM) services to all members of the IBM eServer family, making possible end-to-end workload monitoring in heterogeneous environments, such as a WebSphere Application Server and CICS TS environment. Additionally, IPIC requests originating in a JCA resource adapter will automatically contain point-of-entry information, enabling CICSplex SM (or equivalent CICS monitoring tools) to perform problem determination and offline analysis of requests as they enter and flow across a CICSplex.
- ▶ **Channels as modern-day COMMAREAs:** Interoperation with the CICS TS V3 channels and containers programming model provides an improved method of exchanging data with CICS programs, in amounts that far exceed the 32 KB limit that applies to COMMAREAs and additionally provides an optimized and more structured data interface. Support is provided for both the JCA ECI resource adapter and the CICS TG base classes.
- ▶ **Extended XA support:** Deployment options when using XA support with the CICS ECI resource adapter are expanded to support both IPIC connections when using the CICS TG for multiplatforms in local mode. XA transaction support enables CICS Transaction Server (CICS TS) on z/OS to participate in a global two-phase commit transaction that is initiated in a Java 5 based distributed J2EE V1.4 application server, such as WebSphere Application Server V6.1.
- ▶ **Extended Secure Sockets Layer (SSL) support:** Options for secure intercommunication are enhanced through the exploitation of the CICS TS V3.2 support for SSL/TLS, when using a local mode CICS TG on any supported platform. This feature enables Java clients to use an encrypted connection to a CICS TS V3.2 system, providing for secure transmission of data and optionally for authentication using X509 certificates.

1.1.2 CICS TG application programming interfaces

CICS TG for Multiplatforms provides the following programming interfaces for accessing CICS applications:

- ▶ External Call Interface (ECI)
- ▶ External Presentation Interface (EPI)
- ▶ External Security Interface (ESI)

Note: The CICS TG on z/OS supports only the ECI, because of its reliance on the CICS EXCI function that provides the call interface and connectivity between the Gateway daemon and the CICS TS region.

External Call Interface

The ECI is used for calling COMMAREA-based CICS programs. The COMMAREA is the buffer that is used for passing the data between the client and the CICS server. CICS sees the client request as a distributed program link (DPL) request.

The ECI enables a user application to call a CICS program synchronously or asynchronously. It enables the design of new applications to be optimized for client-server operation, with the business logic on the server and the presentation logic on the client.

An ECI request can be invoked from a Java application using a variety of different interfaces:

- ▶ The `ECIRequest` class that is provided by the CICS TG base classes.
This interface provides a simple interface to the ECI. It is supported in any Java environment (such as an stand-alone application) and provides similar capabilities to the JCA. However, it does not provide the same qualities of service (such as XA transaction support).
- ▶ The Common Client Interface (CCI) that is provided by the CICS ECI resource adapters (`cicseci.rar` or `cicseciXA.rar`).
These classes define a standard architecture for connecting the Java 2 Platform Enterprise Edition (J2EE) platform to a heterogeneous EIS such as CICS. Java applications interact with resource adapters using the Common Client Interface (CCI), which is a common framework of classes extended by each resource adapter to allow communication with a specific EIS.

External Presentation Interface

The EPI is used for invoking 3270-based transactions. A terminal is installed in CICS, and CICS sees the request as running on a remote terminal controlled by the CICS TG.

An EPI request can be invoked from a Java application using one of three different interfaces:

- ▶ The `EPIRequest` class provided by the CICS TG base classes.
This class provides a Java interface to the EPI, and is used for invoking 3270-based transactions. Due to its low-level nature, using it for developing EPI applications requires a strong knowledge of CICS and 3270 data streams.

- ▶ The EPI support classes, which provide high-level constructs for handling 3270 data streams.
A wide range of classes is provided, including AID, FieldData, Screen, Terminal, Map, and MapData. These are used to represent the interface to a CICS 3270 terminal, and the resulting 3270 response.
- ▶ The Common Client Interface (CCI) provided by the CICS EPI resource adapter (cicsepi.rar).

EPI is typically used when it is not possible to separate the presentation logic from the business logic of an application, and allows the reformatting of the user interface, without modification of the CICS application. For a more detailed discussion on the benefits of separating business logic from presentation logic, refer to *Architecting Access to CICS within an SOA*, SG24-5466.

External Security Interface

The ESI is used for verifying and changing the user ID and password information held in the CICS external security manager (ESM), such as RACF®. It is based on the CICS Password Expiration Management (PEM) function.

ESI calls to CICS can only be made using the ESIRRequest class that is provided by the CICS TG base classes. This class provides a Java interface to the ESI, and provides two simple PEM requester functions:

- | | |
|------------------------|--|
| Verify Password | Allows a client application to verify that a password matches the password for a given user ID that is stored by the CICS ESM. |
| Change Password | Allows a client application to change the password that is held by the CICS ESM for a given user ID. |

ESI is supported through the Java Base Classes, although both the EPI and ECI allow user IDs and passwords to be flowed within the actual requests.

1.2 Local and remote modes of operation

There are two different modes of operation when using the CICS TG, remote and local, which are detailed in the following section.

1.2.1 Remote mode of operation

In the *remote mode* of operation, the Gateway daemon runs as a long running task that listens on specified ports for incoming ECI requests and then forwards them to the CICS server through the EXCI or IPIC protocols. The Gateway daemon runs in its own z/OS address space and provides connection and thread management, as shown in Figure 1-2 on page 8.

1.2.2 Local mode of operation

If the CICS TG is to be used on the same machine as WebSphere Application Server, it is more efficient to use the CICS TG classes within WebSphere Application Server to provide the gateway functionality (Figure 1-3). This mode of operation allows WebSphere Application Server to manage the connections and threads and reduces the communications impact. This configuration is known as the *local mode* of operation.

Note: Although local mode may be chosen for its optimal performance, it does not provide the ability to perform any systems monitoring, because this function is provided by the Gateway daemon component.

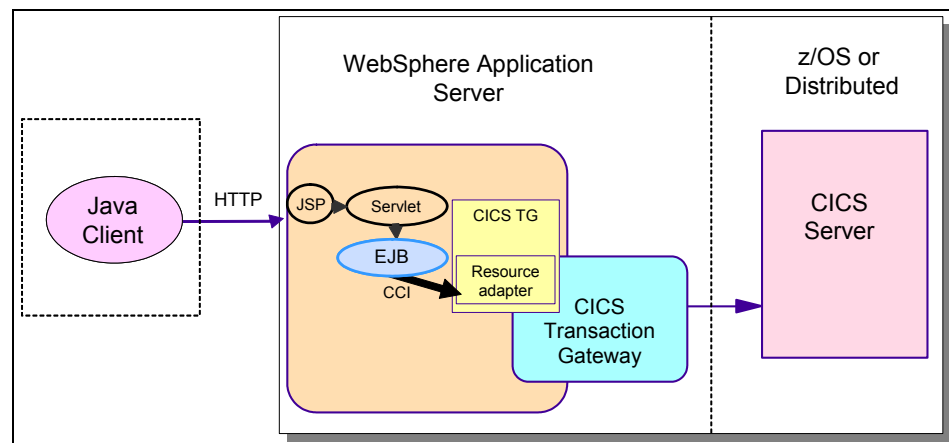


Figure 1-3 CICS TG in local mode with WebSphere Application Server

There is further discussion of the local and remote modes of operation in 1.4, “Using the JCA with different CICS TG topologies” on page 16.

1.3 JCA

The J2EE connector architecture (JCA) defines a standard for connecting the Java 2 Platform, Enterprise Edition (J2EE) to heterogeneous Enterprise Information Systems (EIS) such as CICS. The architecture defines a set of scalable, secure, and transactional mechanisms that enable the integration of EISs with application servers and enterprise applications. A J2EE application server, such as IBM WebSphere Application Server, can be extended to support the resource adapter architecture and is then assured of a seamless connectivity to multiple EISs. Likewise, an EIS vendor provides one standard resource adapter with the capability to plug into any application server that supports the connector architecture.

1.3.1 Overview of JCA

Figure 1-4 shows the key parts of the JCA: the deployable components (resource adapters), the programming interface they implement (CCI), and the qualities of service that they implement (system contracts).

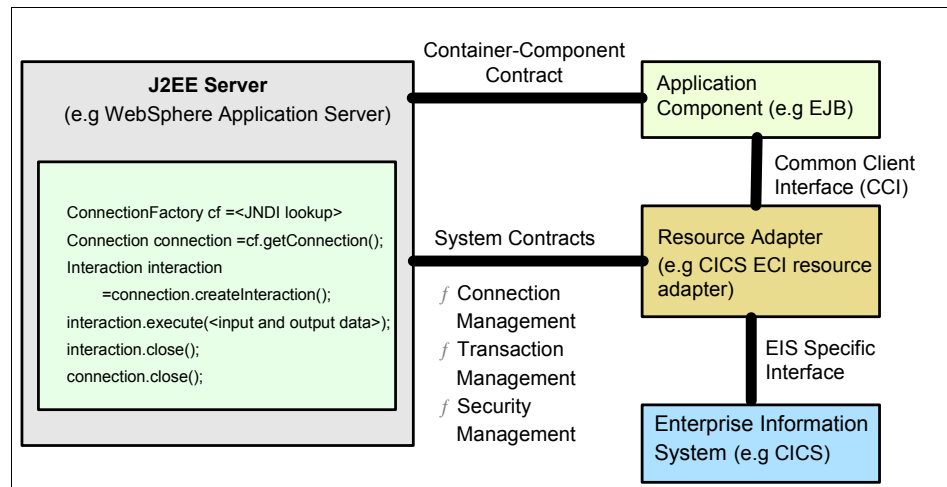


Figure 1-4 J2EE Connector Architecture (JCA)

Resource adapters

A resource adapter is a system-level software driver that a Java application uses to connect to an EIS. A resource adapter is installed into an application server and provides connectivity between the EIS, the application server, and the enterprise application.

CCI

The Common Client Interface (CCI) defines a standard client API for application components to access multiple resource adapters. This API can be used directly, or enterprise application integration frameworks (EAI) can be used to generate EIS access code for the developer. The CCI is designed to be an EIS independent API, such that an enterprise application development tool can produce code for any J2EE compliant resource adapter that implements the CCI interface.

The CCI has the following additional characteristics:

- ▶ It forms a base level API for EIS access on which higher level functionality specific to an EIS can be built.
- ▶ It provides an API that is consistent with other APIs in the J2EE platform, such as JDBC™.
- ▶ It is targeted primarily towards application development tools and enterprise application integration frameworks, rather than Java developers using the CCI API directly.
- ▶ It is an optional part of the JCA specification.

System contracts

The Connector architecture defines a standard set of system-level contracts between a J2EE server and a resource adapter. The standard contracts are as follows:

- ▶ A *connection management* contract that allows J2EE server pool connections to an underlying EIS, and lets application components connect to an EIS. This leads to a scalable application environment that can support a large number of clients requiring access to EIS systems.
- ▶ A *transaction management* contract between the transaction manager and an EIS that supports transactional access to EIS resource managers. This contract lets a J2EE server use a transaction manager to manage transactions across multiple resource managers. This contract also supports transactions that are managed internal to an EIS resource manager without the necessity of involving an external transaction manager.

- ▶ A *security* contract that enables secure access to an EIS. This contract provides support for a secure application environment, which reduces security threats to the EIS and protects valuable information resources managed by the EIS.

These system contracts are transparent to the application developer, meaning they do not have to implement these services themselves.

JCA Version 1.5

In addition to the standard connection, transaction, and security system contracts, JCA Version 1.5 provides four additional system contracts:

- ▶ Life cycle management, allowing an application server to control the startup and termination of a resource adapter. This is not supported by the CICS TG or by WebSphere Application Server.
- ▶ Work management, allowing a resource adapter to submit work instances to an application server for execution. This is not supported by the CICS TG or by WebSphere Application Server.
- ▶ Message inflow, allowing a resource adapter to asynchronously deliver messages to message endpoints residing in the application server.
- ▶ Transaction inflow, allowing a resource adapter to import a transaction context from the EIS system into an application server.

JCA V1.5 also provides two optimizations to the transaction management and connection management contracts, which are exploited in the JCA V1.5 resource adapters provided by CICS TG V7.1:

- ▶ Lazy connection association
This contract provides for potential improved connection pooling in J2EE application servers. It allows the resource adapter to disassociate a cached connection handle from the managed connection once the connection handle is no longer being used by the J2EE component. This allows J2EE applications to use the *get-use-cache* model for connection handles, and for the underlying connections to still be efficiently pooled by the Connection Pool manager in WebSphere Application Server.
- ▶ Lazy transaction enlistment
This contract optimizes transaction enlistment calls so that the resource adapter only participates in a transaction if it is actually invoked, rather than being enlisted whenever a reference exists within the J2EE component. This provides for potential better performance if transactional JCA applications are used in an J2EE application server.

1.3.2 CICS resource adapters

The following resource adapters are provided for use with the CICS TG for Multiplatforms. The resource adapter archives (RAR files) are supplied in the <install_path>/deployable directory.

cicsecl.rar	The CICS ECI resource adapter, which provides one-phase transactional support.
cicsepi.rar	The CICS EPI resource adapter, which is non-transactional.
cicseclXA.rar	The CICS ECI XA resource adapter, which provides two-phase transactional support when deployed into WebSphere Application Server on any supported platform.

The following resource adapters are provided for use with the CICS TG on z/OS. The resource adapter archives (RAR files) are supplied in the <install_path>/deployable directory.

cicsecl.rar	The CICS ECI resource adapter, which provides one-phase transaction support when installed into WebSphere Application Server on a distributed platform, and two-phase transactional support when deployed into WebSphere Application Server on z/OS.
cicseclXA.rar	The CICS ECI XA resource adapter, which provides two-phase transactional support when deployed into WebSphere Application Server on any supported platform.

1.4 Using the JCA with different CICS TG topologies

The qualities of service provided by the JCA vary, depending on the topology in use. The most common topologies in use today are shown in Figure 1-5 on page 17 and are as follows:

- ▶ Topology 1: WebSphere Application Server and the CICS Transaction Gateway are both deployed on a distributed (non-System z) platform.
- ▶ Topology 2: WebSphere Application Server is deployed on a distributed platform and CICS Transaction Gateway is deployed on a z/OS system.
- ▶ Topology 3: Both WebSphere Application Server and CICS Transaction Gateway are deployed on System z™.

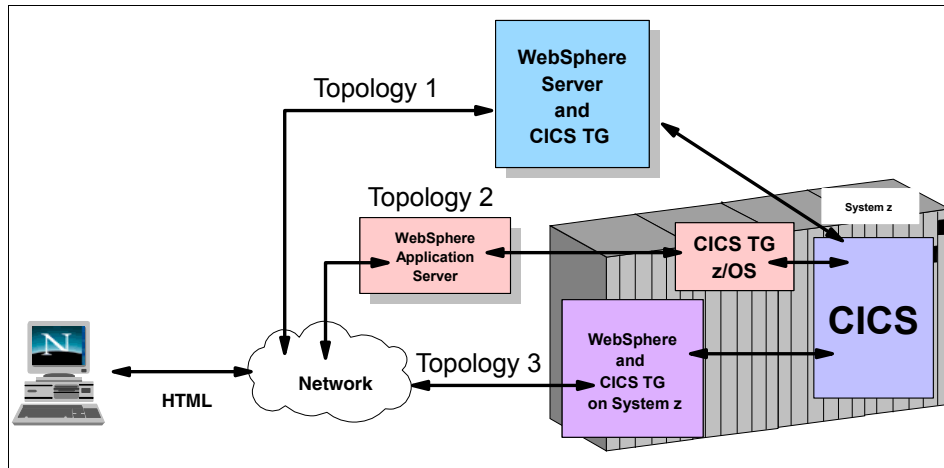


Figure 1-5 JCA deployment topologies

1.4.1 WebSphere Application Server and CICS TG: distributed platform

In this topology, both WebSphere Application Server and CICS TG are deployed on one of the distributed platforms, such as AIX or Linux (Figure 1-6).

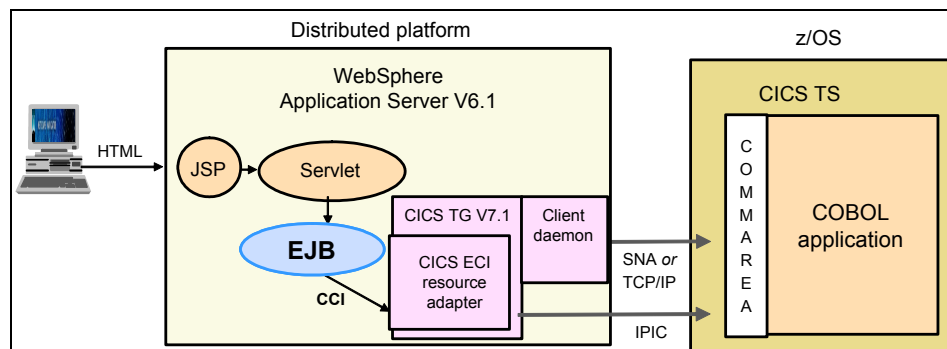


Figure 1-6 WebSphere Application Server and CICS TG: distributed platform

The Gateway daemon is not required in this configuration because the CICS TG is used in local mode to invoke the transport drivers directly from the J2EE enterprise bean. The ECI requests are flowed directly to the CICS server using either a TCP/IP, a Systems Network Architecture (SNA) connection, or an IPIC connection. The management of connections, transactions, and security is controlled within WebSphere Application Server through a combination of both configuration parameters and application deployment descriptors.

The specific qualities of service (in terms of the JCA system contracts) that apply to this topology are as follows.

- Connection pooling

Pooling of connections is provided seamlessly by the pool manager component of WebSphere Application Server allowing the reuse of connections to the resource adapter between multiple J2EE components.

Note: The TCP/IP or SNA network connections from the Client daemon into the CICS region are managed and re-used by the Client daemon component of the CICS TG and are not subject to the JCA connection pooling system contract.

- Transaction management

In this topology, two-phase commit global transactions are supported, using the CICS ECI XA resource adapter and IPIC connections directly into CICS TS V3.2. In addition, the CICS ECI resource adapter (cicseci.rar) can be used with any version of CICS, but it only supports the LocalTransaction interface. As a result, the scope of the transaction is limited to the Resource Manager (that is, the associated connection factory and the specified CICS server). Such Resource Manager LocalTransactions only support one-phase commit processing. However, if the J2EE application server supports the JCA option of *last-resource commit optimization*, an ECI interaction can participate in a global transaction, provided that it is the only one-phase-commit resource in the global transaction. This function is provided by the *last-participant support* in WebSphere Application Server V6.0.

- Security management

Security credentials (user ID and password) propagated through to CICS from WebSphere Application Server can be determined by the application (*component-managed*) or by the Web or EJB™ container (*container-managed*). Container-managed sign-on is recommended because it is good practice to separate the business logic of an application from qualities of service, such as security and transactions. In this topology, however, the principal means of enabling container-managed authentication is by specifying the user ID and password in a *JCA authentication entry* (also known as an alias) and associating the alias with the resource reference when the J2EE application is deployed.

1.4.2 WebSphere Application Server on distributed, CICS TG on z/OS

When WebSphere Application Server is deployed on one of the distributed platforms, it is possible to access CICS through a Gateway daemon running on z/OS (Figure 1-7).

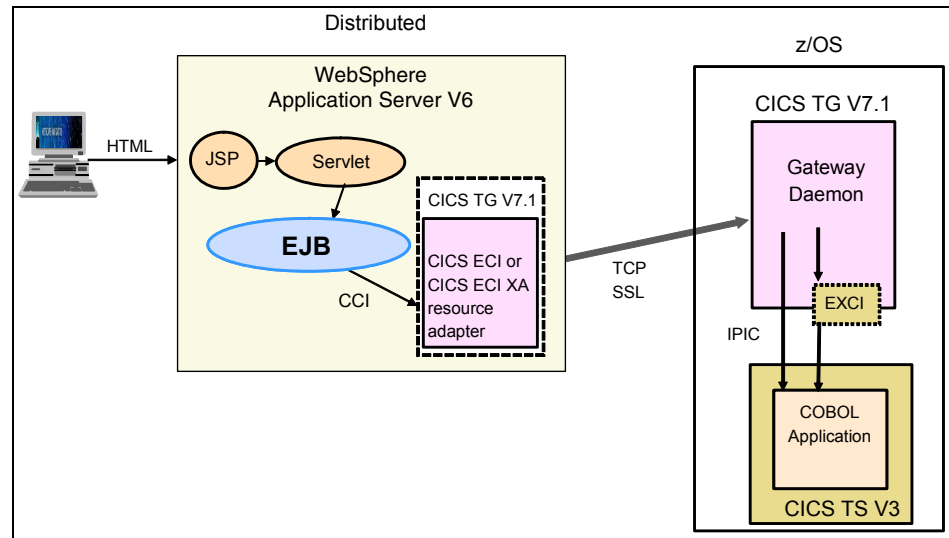


Figure 1-7 WebSphere Application Server on distributed and CICS TG on z/OS

The protocol specified in the connection settings of the connection factory is one of the remote protocols (TCP or SSL). The communication from the Gateway daemon on z/OS to the CICS region uses either the cross memory CICS EXCI protocol, or if using CICS TS V3.2, the TCP/IP socket based IPIC protocol.

This topology enables work to be efficiently distributed across a Parallel Sysplex® using the z/OS internet protocol (IP) workload-management functions, including Sysplex Distributor and TCP/IP port sharing. These technologies allow an individual Gateway daemon to be removed as a single point of failure and enables incoming work from remote clients to be efficiently balanced across multiple Gateway daemons in different z/OS logical partitions (LPARs).

The specific JCA qualities of service that apply to this topology are as follows:

- Connection pooling

The connection pool represents physical network connections between WebSphere Application Server and the Gateway daemon on z/OS. In such a configuration, it is essential to have an efficient connection-pooling mechanism because otherwise a significant proportion of the time from making the connection to receiving the result from CICS and closing the

connection can be in the creation and destruction of the connection itself. The JCA connection-pooling mechanism mitigates this impact by allowing connections to be pooled by the WebSphere Application Server pool manager.

- Transaction management

In this topology, two-phase commit global transactions are supported, using the CICS ECI XA resource adapter and either EXCI or IPIC connections into CICS. Also, one-phase commit transactions using the CICS ECI resource adapter and extended units-of-work are still supported. Note that in either case, if the enterprise bean is deployed with a transactional deployment descriptor (for example, a value of REQUIRED), the resulting ECI request to the CICS region uses transactional EXCI.

- Security management

In this configuration, the Gateway daemon is the entry point to the System in which your CICS system is running, so it is normal for the Gateway daemon to perform an authentication check for incoming ECI requests from clients. However, after the user has authenticated to WebSphere Application Server, a password might not be available to send to the Gateway daemon. In this case, therefore, you need to devise a way to establish a trust relationship between WebSphere Application Server and the Gateway daemon so that WebSphere Application Server can be trusted to flow only the user ID on the request through to CICS Transaction Gateway. Solutions such as SSL client authentication and virtual private networks (VPNs) can be used to establish such a trust relationship.

1.4.3 WebSphere Application Server and CICS TG on System z

In a System z topology, WebSphere Application Server can be deployed on either a z/OS system or on a Linux operating system. The qualities of service differ between these two topologies and are therefore discussed separately.

WebSphere Application Server and CICS TG on z/OS

In this topology (Figure 1-8 on page 21), only the CICS ECI resource adapters are supported. The most common z/OS configuration uses the local mode of operation. This results in a direct cross-memory connection between WebSphere Application Server and CICS, either using EXCI or using optimized fast local sockets. Figure 1-8 on page 21 shows an application deployed to WebSphere Application Server on z/OS.

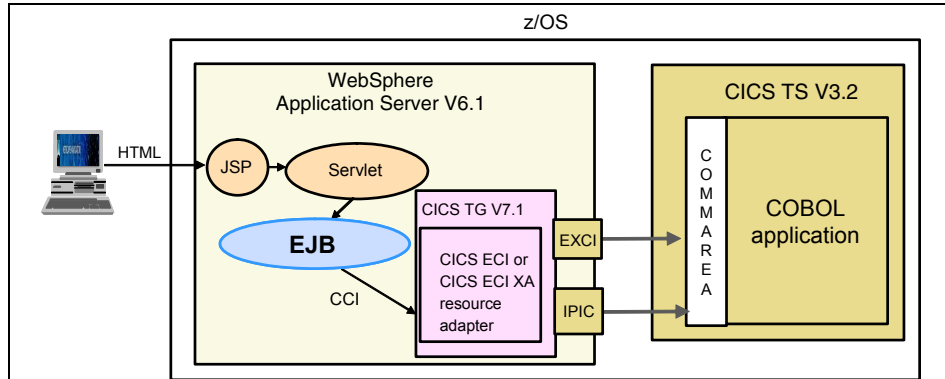


Figure 1-8 WebSphere Application Server and CICS TG on z/OS

Note that the remote mode of operation is also supported, and can be useful if it is necessary to run WebSphere Application Server in a separate LPAR to CICS and the Gateway daemon. However, the highest qualities of service can be achieved when the CICS TG local mode of operation is used.

The specific JCA qualities of service that apply to this topology are as follows:

- Connection pooling

The connection pool is a set of connection objects managed by WebSphere Application Server, which is not directly associated with the EXCI pipes or IPIC sessions used for communication to CICS.

- Transaction management

A two-phase commit capability is provided for either EXCI or IPIC connections. When using EXCI, RRS acts as the external transaction coordinator managing the transaction scope between WebSphere Application Server and CICS. However, when using IPIC XA, requests are sent directly to CICS.

- Security management

Both container-managed and component-managed sign-on are supported. In this topology, the ECI resource adapters flow only the user ID to CICS with the ECI request; it is assumed that the user is already authenticated by WebSphere Application Server. When using container-managed sign-on, a z/OS specific functionality known as *thread identity support* is provided by WebSphere Application Server on z/OS.

Note: *Thread identity support* allows an authenticated RACF identity from WebSphere to be propagated through the CICS TG into CICS for each request.

CICS TG for Linux on System z

WebSphere Application Server and CICS TG deployed on System z (Figure 1-9) provides a flexible and scalable environment based on the virtualization capabilities of IBM z/VM® and Linux systems.

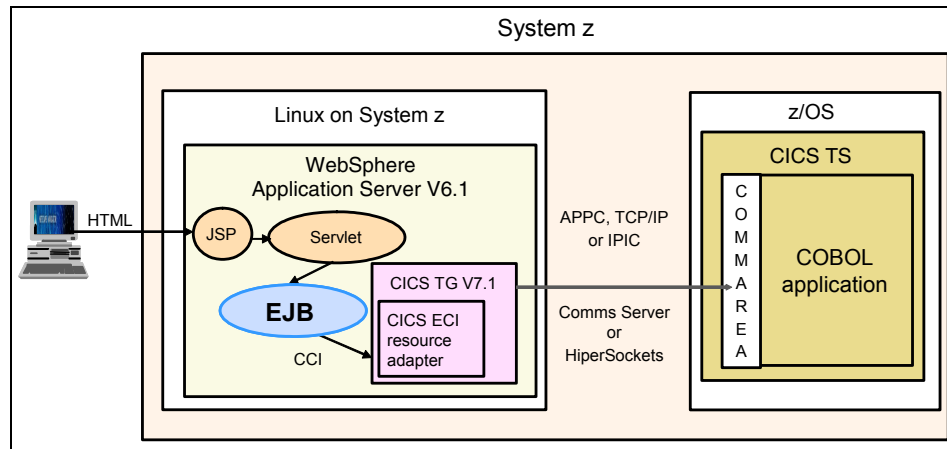


Figure 1-9 WebSphere Application Server and CICS TG for Linux on System z

The JCA qualities of service for this topology are almost identical to those described in 1.4.1, “WebSphere Application Server and CICS TG: distributed platform” on page 17 because Linux on System z (within a JCA and CICS TG scenario) can be treated as a distributed platform. A significant exception to this generalization is that the HiperSockets™ function can be used to provide a highly efficient cross-memory transport for TCP/IP based communication into CICS (using either the ECI over TCP/IP function of CICS TS V2.2 and later releases or the IPIC function of CICS TS V3.2). Alternatively, an APPC connection to CICS TS on z/OS or VSE can be provided by IBM Communications Server for Linux.

1.4.4 Mixed version support

The client-server nature of the CICS TG means that different levels of the Gateway daemon and the Java client (including the resource adapter) can interoperate. This provides a degree of flexibility in deployment, allowing software components to be updated at different times. However, because of the defined internal flow levels and of the JNI™ interface, you should follow these rules to ensure interoperability:

1. Remote Java clients connecting to a Gateway daemon must use a CICS resource adapter of the same (or lower) level CICS TG version as the Gateway daemon.

2. If the CICS TG is used in local mode within WebSphere Application Server, then the Java client and the CICS TG JNI library must be at the same CICS TG level.
3. All components deployed into a Java based runtime must be built using an SDK at the same or lower level as that of the runtime (which means that Java 5 built components such as the CICS TG V7.1 will only function in Java 5 based application servers such as WebSphere Application Server V6.1).

Full details of the latest supported software combinations are provided on the ibm.com support pages for the CICS TG at:

<http://www-1.ibm.com/support/docview.wss?uid=swg21239203>

Here are some examples of supported configurations:

- ▶ Using a Java5 based JCA V1.5 resource adapter (from CICS TG V7.1 or CICS TG V7.0) deployed in WebSphere Application Server V6.1 to connect to a CICS TG V7.1 Gateway daemon is supported.
- ▶ Using a JCA V1.5 resource adapter (from CICS TG V6.1 or CICS TG V6.0) deployed in WebSphere Application Server V6 to connect to a CICS TG V6.1 Gateway daemon is supported.
- ▶ Using a JCA V1.0 resource adapter (from CICS TG v5.1) deployed in WebSphere Application Server V5.1 to connect to CICS TG V6.1 Gateway daemon is supported.
- ▶ Using a JCA V1.0 resource adapter (from CICS TG v5.1) deployed in WebSphere Application Server V6.1 to connect to a CICS TG V5.1 Gateway daemon is supported.

Here are some examples of non-supported configurations:

- ▶ Using a JCA V1.5 resource adapter (from CICS TG V6.0) deployed in WebSphere Application Server V5.1 is not supported. This is because JCA V1.5 is not supported in J2EE 1.3 based application servers such as WebSphere Application Server V5.1
- ▶ Using a Java 5 built resource adapter from CICS TG V7.0 deployed in WebSphere Application Server V6.0 is not supported. This is because WebSphere Application Server V6.0 is a Java 1.4 based runtime and as such does not support components built with Java 5.
- ▶ Using a resource adapter from CICS TG V6.0 deployed in WebSphere Application Server V6 to connect to a CICS TG V5.1 Gateway daemon is not supported. This is because the resource adapter is at a later level than the Gateway daemon.

- Using a CICS TG V5.1 JNI library for use in local mode with a CICS TG V6.0 resource adapter deployed in WebSphere Application Server V6.0 is not supported. This is because the CICS TG resource adapter and JNI library must always be at the same level for use in local mode.

Note: Do not mix the levels of CICS resource adapters that are deployed on a WebSphere Application Server node. For example, do not install a resource adapter from CICS TG V6.1 to a node that has a V6.0 ECI or EPI resource adapter installed. The resource adapter version is shown in the ra.xml file in the resource adapter.



Configuring the CICS TG

In this chapter, we describe how to install and configure the CICS Transaction Gateway on z/OS V7.1 and how to test the installation with one of the sample Java applications that is supplied with the CICS TG on z/OS.

2.1 Preparing for the installation

The following sections detail the software levels that we use in our configuration and include instructions on how to install the CICS TG on z/OS.

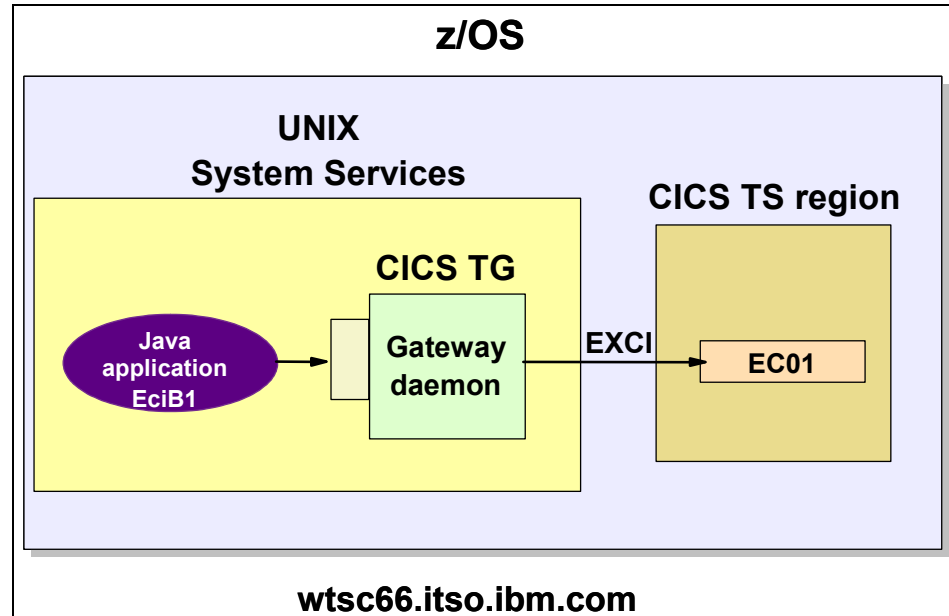


Figure 2-1 Software components: CICS TG on z/OS

The CICS TG runs in the z/OS UNIX® System Services environment. You need a mix of traditional MVS™ skills and UNIX skills in order to install and to configure the CICS TG.

2.1.1 Software checklist

We use the following software levels:

- ▶ z/OS V1R8
- ▶ CICS Transaction Gateway on z/OS V7.1
- ▶ IBM SDK on z/OS, Java 2 Technology Edition, V1.5
- ▶ IBM 31-bit Runtime Environment on z/OS, Java 2 Technology Edition, Version 5.0
- ▶ CICS Transaction Server on z/OS V3.2

The minimum levels that are supported with CICS TG V7.1 are:

- ▶ z/OS V1R7
- ▶ IBM SDK on z/OS, Java 2 Technology Edition, V1.5
- ▶ CICS Transaction Server on z/OS V2.2

2.1.2 Basic test configuration

The basic configuration (see Figure 2-2) for our testing consisted of identical CICS TS regions connected to two CICS Transaction Gateways. This provided a high availability environment capable of enduring a component failure or system slowdown. One of our aims was to monitor workloads under these conditions.

The first part of this chapter focuses on how to install and configure a single CICS Transaction Gateway (SCSCT711). Section 2.6.3, “Configuring for multiple Gateway daemons” on page 65 will explain the steps necessary to create a second (SCSCT714) Gateway daemon and possibly subsequent Gateway daemons should we need more. Decisions about naming conventions and design of the Hierarchical File System (HFS) structure were based on this multi-component configuration.

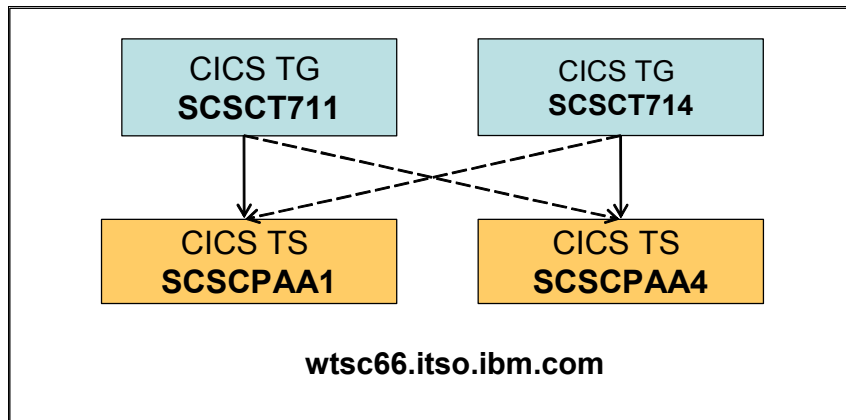


Figure 2-2 Basic test configuration

2.1.3 Definitions checklist

Table 2-1 summarizes the CICS definitions used in this scenario. Before you configure the products, we recommend that you acquire definitions for each value that is listed in this table.

Table 2-1 Definitions checklist

Value	SCSCT711	SCSCPAA1
Host name	wtsc66.itso.ibm.com	-
IP address	9.12.4.75	-
TCP/IP port	2006	-
Job name	SCSCT711	SCSCPAA1
APPLID	SCSCT711	SCSCPAA1
NETNAME	SCSCT711	SCSCPAA1
CONNECTION	-	T711 (SCSCT711) T114 (SCSCT714)

2.2 Installing CICS TG

To install the product, you need to perform the following steps:

1. Install product files.
2. Consider basic security setup.

2.2.1 Installing product files

During the SMP/E installation of CICS TG, the HFS data set CTG.V7R1M0.SCTGHFS that contains the CICS TG product files was created and mounted at the mount point `usr/lpp/cicstg/ctg710` (known as the `<install_path>`). For more information about the SMP/E installation, see Program

Directory for CICS Transaction Gateway on z/OS, GI10-2587. Figure 2-3 shows the resulting directory structure that follows the installation of CICS TG.

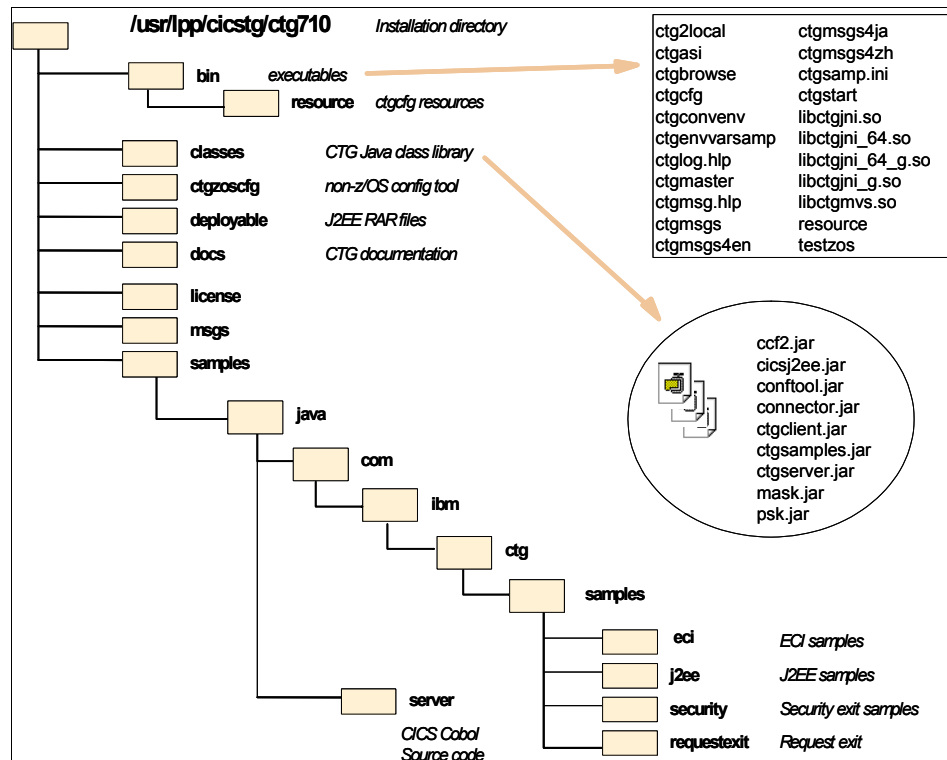


Figure 2-3 CICS TG on z/OS directory structure

Note: All the directories have an IBM directory that contains files with names in the form CTGnnnnn, where nnnnn are digits. These directories and files are part of the SMP/E installation. You should not alter or remove them.

The /usr/lpp/cicstg/ctg710/bin directory contains the following source sample files, which are referred to later in this chapter:

ctgsamp.ini	Sample CICS TG configuration file
ctgenvvarsamp	Sampled ctgenvvar script
ctgstart	Shell script to start the Gateway daemon

Tip: You can browse files using TSO OBROWSE or ISHELL commands.

The /usr/lpp/cicstg/ctg710/classes directory contains the following JAR files:

ctgclient.jar	Java class library
ctgserver.jar	Classes for use by the Gateway daemon
ctgsamples.jar	Samples
cicsj2ee.jar, ccf2.jar, connector.jar	J2EE classes
mask.jar, psk.jar, guide.jar	Classes used by the configuration tool

Tip: The contents of a JAR or zipped file can be listed with the UNIX System Services command in OMVS:

```
jar -tvf <file>
```

2.2.2 Basic security considerations

You must perform the following basic RACF security tasks before starting the Gateway daemon:

1. Set up a user ID for the Gateway daemon started task.
2. Allow the Gateway daemon user ID read access to program-controlled libraries.
3. Define programs and libraries as program-controlled.

Setting up an user ID for the Gateway daemon started task

The user ID under which the Gateway daemon started task runs should:

- ▶ Have an OMVS segment defined.
- ▶ Be in a group that has an OMVS segment.
- ▶ Be defined without a password.
- ▶ Have READ access to the RACF profile that protects the TCP/IP.STANDARD.TCPXLBIN data set.

We ran our Gateway daemon as a started task under user ID CTGUSER.

There are two methods of defining a default user ID to a started task:

- Use the RACF exit - ICHRIN03.
- Use the STARTED class in RACF.

The following example shows the user ID CTGUSER in the CICS group being assigned as the user ID for the SCSTG711 task:

```
RDEF STARTED SCSTG71* STDATA(USER(CTGUSER) PRIVILEGED(NO)
TRUSTED(NO))
```

After the RDEF command was issued, the started task continued to use the default user ID until the following command was issued:

```
SETROPTS RACLIST(STARTED) REFRESH
```

Allow Gateway daemon access to program-controlled MVS libraries

If the z/OS system has the BPX.SERVER FACILITY class profile defined within RACF, then the user ID under which the Gateway daemon runs must be permitted to this profile. The following example shows the PERMIT command that we issue for our CTGUSER Gateway daemon user ID:

```
PERMIT BPX.SERVER CLASS(FACILITY) ID(CTGUSER) ACCESS(READ)
PERMIT BPX.FILEATTR.PROGCTL CLASS(FACILITY) ID(CTGUSER) ACCESS(READ)
```

If the BPX.SERVER FACILITY class is not defined, the Gateway daemon user ID must be defined with a UID of 0 (that is, be a root user).

Defining programs and libraries as program controlled

UNIX System Services program control allows RACF to secure UNIX System Services executables as though they were MVS programs. If a nonprogram controlled program is loaded into the Gateway daemon, the address space becomes *dirty*, and program control is lost, which results in security failures in the Gateway daemon.

Tip: The `ls -E` command from an OMVS screen shows the extended attributes of HFS file, including the program control attribute `p`.

After the SMP/E installation of CICS TG, we noted that the HFS files that need to be program controlled had been marked with the extended attribute `+p`.

Tip: The **ctgstart** script is installed with the shared bit *s* set (Example 2-1) and should be left with this setting. This setting causes the **_BPX_SHAREAS** environment variable to be used, which allows the required sharing of the address space with CTGBATCH and non-sharing of the address space when using **ctgstart** from UNIX System Services.

Example 2-1 Results of the ls -E command showing the extended attributes of ctgstart

```
CICSRS6:/pp/ctg710/bin: >ls -E
total 8704
drwxrwxr-x      2 HAIMO   SYS1          928 Sep 24 20:45 IBM
.....
-rwxrwxr-x --s-  2 HAIMO   SYS1        47138 Sep 24 20:45 ctgmsgs
-rwxrwxr-x --s-  2 HAIMO   SYS1        26557 Sep 24 20:45 ctgmsgs4en
-rwxrwxr-x --s-  2 HAIMO   SYS1        26557 Sep 24 20:45 ctgmsgs4ja
-rwxrwxr-x --s-  2 HAIMO   SYS1        26557 Sep 24 20:45 ctgmsgs4zh
-rwxrwxr-x --s-  2 HAIMO   SYS1        12297 Sep 24 20:45 ctgsamp.ini
-rwxrwxr-x -ps-  2 HAIMO   SYS1       38870 Sep 24 20:45 ctgstart
-rwxrwxr-x -ps-  2 HAIMO   SYS1     958464 Sep 24 20:45 libctgjni.so
-rwxrwxr-x -ps-  2 HAIMO   SYS1     548864 Sep 24 20:45 libctgjni_64.so
-rwxrwxr-x -ps-  2 HAIMO   SYS1     548864 Sep 24 20:45 libctgjni_64_g.so
-rwxrwxr-x -ps-  2 HAIMO   SYS1     958464 Sep 24 20:45 libctgjni_g.so
-rwxrwxr-x -ps-  2 HAIMO   SYS1     131072 Sep 24 20:45 libctgmvs.so
drwxrwxr-x      4 HAIMO   SYS1         2240 Sep 24 20:45 resource
-rwxrwxr-x --s-  2 HAIMO   SYS1         8192 Sep 24 20:45 testzos
```

If the files are subsequently modified, they lose their program controlled status and might need to be reset, as follows:

```
extattr +p <install_path>/bin/lib*.so
extattr +ps <install_path>/bin/ctgstart
```

The Java SDK must also be program controlled. This option is set by default when the Java SDK is installed.

The Gateway daemon requires files loaded from MVS libraries. Of these libraries, SCTGLOAD, SDFHEXCI, SDFHLINK, and SCEERUN2 must be program controlled. To give these libraries PROGRAM CONTROL status, issue the following RACF commands:

```
RALTER PROGRAM * ADDMEM('CTG.V7R1M0.SCTGLOAD'//NOPADCHK)
RALTER PROGRAM * ADDMEM('CICSTS32.CICS.SDFHEXCI'//NOPADCHK)
RALTER PROGRAM * ADDMEM('CICST32C.CICS.SDFHLINK'//NOPADCHK)
RALTER PROGRAM * ADDMEM('CEE.SCEERUN2'//NOPADCHK)
```

Then, make the program control settings active with the following command:

```
SETROPTS WHEN(PROGRAM) REFRESH
```

You can find more information about the RACF commands in the *Security Server RACF System Programmer's Guide*, SA22-7681.

2.3 Configuring CICS TG

In this section, we describe how to configure CICS TG. To configure CICS TG, you need to:

1. Define a HFS.
2. Set directory permissions.
3. Create a Gateway daemon configuration file.
4. Create an STDENV file.
5. Create the started task JCL.
6. Configure CICS TG for RRMS.
7. Configure CICS TG for SMF recording.

2.3.1 Defining an HFS

In this section, we show how we created and mounted HFS data sets to hold CICS TG trace and configuration data.

HFS Directory layout

In our environment, we did not want the configuration and trace files to be written to the same HFS as the CICS TG product executables, which we mounted with read only access.

As our environment consists of multiple CICS TGs, we needed to create separate trace and configuration files for each Gateway daemon. We choose to place each Gateway daemon in its own HFS structures (see Figure 2-4 on page 34). Also, we placed the trace files (ctg.trc and jni.trace) in their own directory rather than on the CICS TG *tmp* directory. This would avoid the situation where trace data could fill tmp and prevent the CICS TG from starting.

Table 2-2 and Figure 2-4 shows the Gateway daemon HFS directory layout, including the MVS dataset mount points.

Table 2-2 HFS datasets and their mountpoints

HFS	Mount point
CTGUSER.SCSCSTG71.HFS	/ctg/scscstg71/
CTGUSER.SCSTG711.HFS	/ctg/scscstg71/scstg711/
CTGUSER.SCSTG711.TMP.HFS	/ctg/scscstg71/scstg711/tmp/
CTGUSER.SCSTG714.HFS	/ctg/scscstg71/scstg714/
CTGUSER.SCSTG714.TMP.HFS	/ctg/scscstg71/scstg714/tmp/

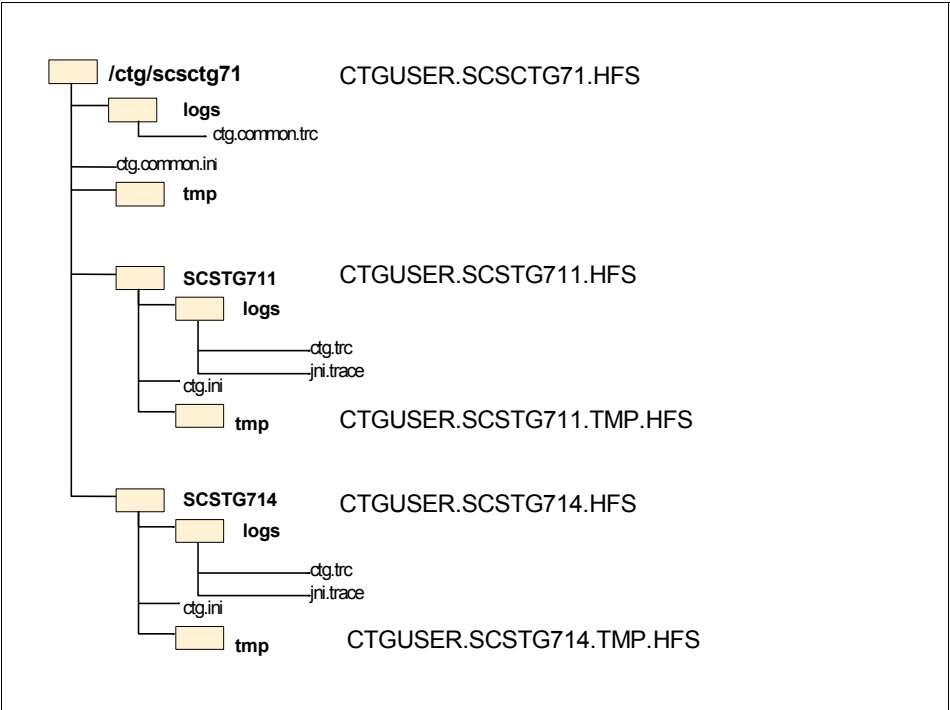


Figure 2-4 CICS TG HFS directory layout

Creating a data set

An HFS can be allocated using ISPF option 3.2, an IEFBR14 batch job, or through the ISHELL menus.

zSeries® File System (zFS) has improved caching functionality compared with HFS. This improved caching can provide significant performance benefits for log and trace files in a zFS compared with equivalent logs on an HFS. Therefore, we used zFS for ctg/scsctg71/scstg71 for the logs and HFS for ctg/scsctg71/scstg711/tmp for the Gateway daemon tmp directory. This book does not discuss zFS. For information about zFS, see *z/OS V1R8.0 Distributed File Service zFS Administration*, SC24-5989.

Tip: To view the attributes of the zFS and HFS datasets, issue an IDCAMS command, such as “LISTCAT ENTRIES('CTGUSER.SCSTG712.HFS.DATA') all”, as a TSO command.

Mounting the HFS

In OMVS, change to the root directory and use the OMVS **mkdir** command to create the /ctg/ctg710/scstg711 and /ctg/ctg710/scstg711/tmp directories (Example 2-2).

Example 2-2 Creating a mount point directory

```
CICSRS6: />mkdir ctg
CICSRS6: />mkdir ctg/scsctg71
CICSRS6: />mkdir ctg/scsctg71/scstg711
CICSRS6: />mkdir ctg/scsctg71/scstg711/tmp
```

Then, mount each of the HFS data sets using the ISHELL File_systems menu (see Figure 2-5).

File Directory Special_file Tools File_systems Options Setup Help

Mount a File System

Mount point:

More: +

/ctg/scsctg71/

File system name CTGUSER.SCSTG71.HFS

File system type HFS New owner

Owning system Character Set ID

Select additional mount options:

Read-only file system Set automove attribute...

Ignore SETUID and SETGID Text conversion enabled

Bypass security

Mount parameter:

Command ==> SEL 0

Figure 2-5 Using ISHELL to mount CTGUSER.SCSTG71.HFS

Tip: Issuing the `df .` command on a UNIX System Services command line gives information about the mount table entry for the current HFS directory (see Example 2-3).

Example 2-3 Using the `df .` command

```

CICSR6:/ctg/scsctg71/scstg711/tmp: >df .
Mounted on      Filesystem              Avail/Total    Files      Status
/ctg/scsctg71/scstg711/tmp (CTGUSER.SCSTG711.TMP.HFS) 21276/21600  4294967270
Available
CICSR6:/ctg/scsctg71/scstg711/tmp:>

```


2.3.2 Setting directory permissions

To be able to read the configuration file, the Gateway daemon user ID needs to have:

- ▶ Execute permission on the directories (to traverse the directories)
- ▶ Read permission on the configuration file

In our testing, we planned to have a number of Gateway daemons, all running under user ID CTGUSER. In this environment, each daemon reads its configuration file and writes trace output to the trace directory. To enable this, we connected CTGUSER to the RACF group ID associated with our user ID (87679), which was made the owning group of the following directories:

- ▶ /ctg
- ▶ /ctg/scsctg71
- ▶ /ctg/scsctg71/scstg711
- ▶ /ctg/scsctg71/scstg711/tmp

We use the ISHELL, specifying the relevant directory and selecting **Directory** → **Attributes** → **Edit** → **Owning Group**, as shown in Figure 2-6.

The screenshot displays the ISHELL command-line interface. At the top, a menu bar includes 'File', 'Directory', 'Special_file', 'Tools', 'File_systems', 'Options', 'Setup', and 'Help'. Below this, a sub-menu is open with 'Edit' and 'Help' options. The main display area shows 'Display File Attributes' for the path '/ctg/scsctg71/'. A list of file attributes is visible on the left, including 'File typ', 'Permissi', 'Access c', 'File siz', 'File own', 'Group ow', 'Last mod', 'Last cha', 'Last acc', 'Created', and 'Link cou'. A dialog box titled 'Change the Owning Group' is centered, prompting the user to 'Change the GID or group name and press Enter.'. It contains two input fields: 'GID number 87679' and 'Group name CICS'. To the right of the dialog, there are three horizontal lines and a '+' sign. At the bottom of the screen, the command prompt shows 'Command ==> SEL A'.

Figure 2-6 Changing the GID with ISHELL

We then changed the permissions for the directories to 660 and gave the configuration file /ctg/scsctg71/scstg711/ctg.ini permissions of 660. Failure to grant the right permission will result in error messages being issued at start-up (see “Error with bad permissions on the directory” on page 77).

2.3.3 Creating a Gateway daemon configuration file

CICS TG uses a Gateway daemon configuration file for its configuration parameters. The default name for this configuration file is ctg.ini. A sample configuration file is supplied in the /usr/lpp/cicstg/ctg710/bin/ctgsamp.ini file.

In our testing, we decided to use the same configuration file name for each Gateway daemon, namely *ctg.ini*. Each CICS TG would have its own configuration file located in its directory /ctg/scsctg71/scstg711/. To create our configuration file, we:

- ▶ Copied the sample configuration file /usr/lpp/cicstg/ctg710/bin/ctgsamp.ini to /ctg/scsctg71/scstg711/ctg.ini
- ▶ Edited ctg.ini, as shown in (Example 2-4)

Example 2-4 Configuration file ctg.ini

```
SECTION GATEWAY
initconnect=500
maxconnect=500
initworker=100
maxworker=100
#trace=on
#notime=on
noinput=on
nonames=on
uowvalidation=off
tfile=/ctg/scsctg71/scstg711/logs/ctg.trc
xasupport=on
statsport=2981
healthreporting=on
healthinterval=1
statsrecording=on
statint=010000
protocol@tcp.handler=com.ibm.ctg.server.TCPHandler
protocol@tcp.parameters=port=2006;\
                        connecttimeout=2000;\
                        idletimeout=600000;\
                        pingfrequency=60000

workertimeout=1000
closetimeout=5000
```

```
connectionlogging=on
log@info.dest=console
log@error.dest=console
```

```
ENDSECTION
SECTION PRODUCT
applid=SCST711
ENDSECTION
```

We left most of the settings as their default values. Changes were made the following parameters:

- ▶ maxconnect was set to 500, and initconnect and initworker were set to equal their respective max values. This would cause the ConnectionManager and Worker threads to be initialized at startup, preventing thread allocation processing from affecting our tests. Pre-allocating the threads would also ensure that we discover on startup whether the amount of storage we have allocated is sufficient to accommodate the maximum number of threads, thus avoiding a storage shortage condition during our testing.
- ▶ tfile was changed to point at the logs directory.
- ▶ statsport was changed to the port number where the Gateway daemon would listen for Statistics API requests. This setting is new for CICS TG v7.0.
- ▶ healthrecording was set on to allow the Gateway daemon to report the health of communications with CICS to a TCP/IP load balancer. The load balancer can then use this measure of health to set priorities when creating new incoming IP socket connections to Gateway daemons in the load balancing group.
- ▶ healthinterval was set to one second. The Gateway daemon will wait for one second before checking the health of communications with CICS and report this to the TCP/IP load balancer.
- ▶ Applid is used to identify the CICS Transaction Gateway in the SMF header record in field SMF111_SMFSPN and messages on the z/OS console log. We specified our name for the Gateway daemon, SCST711.

We intend to implement CICS TG transactional support in some of our tests, therefore xasupport=on was also specified. For further details, see 2.3.6, “Configuring CICS TG for RRMS” on page 46 and Chapter 4, “Scenario environment” on page 121 for the WebSphere Application Server configuration.

For more detail about these parameters, see *CICS Transaction Gateway on z/OS Administration*, SC34-6672.

2.3.4 Creating an STDENV file

Before the Gateway daemon can be started, you need to set some environment variables in an STDENV file, which can be an MVS sequential file, a PDS member, or an HFS file. A sample STDENV is supplied in member CTGENV of SCTGSAMP. See 2.7.1, “Migrating from ctgenvar” on page 69 for details of how the **ctgconvenv** script can be used to migrate **ctgenvar** settings into STDENV.

Tip: We recommend that you use a PDS or PDSE member for your STDENV. We found that an STDENV allocated as a sequential data set cannot be updated while the Gateway daemon is running.

You need to set the following environment variables for the CTG:

- ▶ CICSCLI
- ▶ CLASSPATH
- ▶ CTG_JNI_TRACE
- ▶ PATH
- ▶ CTG_RRMNAME
- ▶ DFHJVPIPE
- ▶ DFHJVSYSTEM
- ▶ TMPDIR

We created a STDENV member in CICSSYSF.CTG71.STDENV based on the supplied sample from SCTGSAMP. The name of the STDENV PDS member was SCSC711, the CTG NETNAME. Example 2-5 shows the environment variable settings that we used.

Example 2-5 CICSSYSF.CTG71.STDENV(SCSC711)

```
#ENV VARIABLES FOR CTG PROC SCSC711
_BPXX_SETIBMOPT_TRANSPORT=TCPIP
_BPX_SHAREAS=YES
_CEE_DMPTARG=/ctg/scsctg71/scstg711/logs
AUTH_USERID_PASSWORD=NO
CICSCLI=/ctg/scsctg71/scstg711/ctg.ini
CLASSPATH=/usr/lpp/cicstg/ctg710/classes/ctgsamples.jar
CTG_JNI_TRACE=/ctg/scsctg71/scstg711/logs/jni.trace
COLUMNS=132
DFHJVSYSTEM_00=SCSCPAA1-CICS 3.2
DFHJVSYSTEM_01=SCSCPAA4-CICS 3.2
DFHJVPIPE=SCSC711
PATH=/bin:/usr/lpp/java/J5.0/bin
TMPDIR=/ctg/scsctg71/scstg711/tmp
TZ=EST5EDT
CTGSTART_OPTS=-j-Xshareclasses:name=scsctg71
CTG_PIPE_REUSE=ONE
```

Description of the environment variables

The following list provides a description of the main environment variables and the values that we set:

► **AUTH_USERID_PASSWORD=NO**

This variable is used to control whether the Gateway daemon performs user authentication on the flowed user IDs. We disabled user authentication because we were not using security in our test systems.

► **_BPXK_SETIBMOPT_TRANSPORT=TCPIP**

This is a Communications Server variable that controls the TCP/IP stacks. On our LPAR, two TCP/IP stacks were available, and we used this variable to restrict usage to the stack called TCPIP. The names of the TCP/IP stacks available can be displayed using the MVS system command /D TCPIP.

► **_BPX_SHAREAS**

The Gateway daemon can be started in its own address space or share the address space of the process that started it. If you are using CTGBATCH to start the Gateway daemon (see 2.6.1, “Starting the Gateway daemon” on page 62), ensure that **_BPX_SHAREAS=YES** is set in the STDENV DD statement. If starting the Gateway daemon from a UNIX System Services shell prompt (such as OMVS), set **_BPX_SHAREAS=NO** in the **ctgenvvar** script to force the use of a clean address space.

We specified **_BPX_SHAREAS=YES** because we started the Gateway daemon using CTGBATCH and used one address space.

► **_CEE_DMPTARG**

Set this to the location on the HFS that the JVM will write dumps to.

_CEE_DMPTARG controls the location of javacore, heapdump, snap trace, and transaction dumps. If **_CEE_DMPTARG** is not set, the following list defines how the JVM decides where to send dump files:

- The current working directory of the JVM processes.
- The location specified by the TMPDIR environment variable, if set.
- If the Javadump cannot be stored in any of the above, it is put to STDERR.

► **CICSCLI**

You can specify a different path to the configuration file by setting the location of the configuration file. This variable is important when running multiple Gateway daemon started tasks.

We set CICSCLI to CICSCLI=/ctg/scsctg71/scstg711/ctg.ini to allow each Gateway daemon started task to have its own configuration file.

► CLASSPATH

Includes a concatenation of fully qualified jar files or HFS directories that contain class files. If running the Gateway daemon with the sample security exits, include <install_path>/classes/ctgsamples.jar in the CLASSPATH. The **ctgstart** script appends the main product jar files to the classpath.

We set CLASSPATH to
CLASSPATH=/usr/lpp/cicstg/ctg710/classes/ctgsamples.jar.

► CTG_PIPE_REUSE=ONE

This variable ensures that EXCI pipes are not overallocated, and only one pipe is ever allocated per Worker thread.

► CTG_RRMNAME

This is the name with which the Gateway daemon registers with Resource Recovery Services (RRS) for transactional EXCI requests to CICS. The CTG_RRMNAME must conform to the naming rules for RRS groups.

The name can consist of the following printable characters:

- Alphanumeric characters: A-Z and 0-9
- National characters: \$ (X'5B'), # (X'7B'), @ (X'7C')
- The period (.)
- The underscore (_)

RRM name restrictions include:

- The name cannot start with a blank or contain embedded blanks.
- Lowercase characters are converted to uppercase characters.
- To avoid naming conflicts, use A through C or G through I as the first character.
- The length of CTG_RRMNAME must not exceed 32 characters and must end with the characters IBM.UA.

If you do not specify a CTG_RRMNAME, a unique CTG_RRMNAME is generated by the CICS TG.

We set CTG_RRMNAME to CTG_RRMNAME=SCSCT711.IBM.UA (see 2.3.6, “Configuring CICS TG for RRMS” on page 46).

► CTGSTART_OPTS=-j-Xshareclasses:name=scsctg71

The CTG start options are used to pass parameters to the JVM. The XShareclasses parameter is the name of the shared class cache used by the JVM to store the CICS TG Java byte codes. Use of the Java shared class

cache provides improved startup times and reduced region storage when using cloned Gateway daemons.

► COLUMNS

Specifies the output width of the screen for CICS TG messages. We left this at the COLUMNS=80 entry supplied in the sample STDENV.

► DFHJVPIPE

The value specified here must match the NETNAME in the CICS connection definition to be used for an EXCI connection using a specific pipe. The default connection name in the IBM-supplied group DFH\$EXCI is EXCS, which has a NETNAME of BATCHCLI.

We created our own connection definition, and used a NETNAME of SCSC711.

► DFHJVSYSTEM

The syntax of this variable is DFHJVSYSTEM_*nn=aaaaaaaa literal*, where:

- *nn*: 0 to 99.
- *aaaaaaaa*: The APPLID of the CICS region.
- *literal*: A description of the CICS region.

Note the setting of this variable is optional, and only affects the result of the ECIRRequest.listSystems() method and does not affect which CICS regions to which you can connect.

We set DFHJVSYSTEM_00 to SCSCPAA1, the default CICS TS system.

We set DFHJVSYSTEM_01 to SCSCPAA4, the failover CICS TS system.

► PATH

The PATH statement includes the binary directory where Java is located.

We set PATH to PATH=/bin:/usr/lpp/java/J5.0/bin.

► STEPLIB

This variable identifies the libraries containing EXCI options. It also identifies the EXCI load libraries. If a modified EXCI options table DFHXCOPT is to be included, it must appear before the EXCI load library SDFHEXCI in order to override the default DFHXCOPT table.

We did not set this variable because we specified a STEPLIB in our started task JCL for CICSTS32.CICS.SDFHEXCI.

► TMPDIR

TMPDIR is a UNIX System Services variable defining a temporary directory that is used while executing OMVS commands.

The default location for this directory is /tmp. We kept the TMPDIR within the CICS TG HFS in directory /ctg/scscctg71/scstg711/tmp.

► TZ

TZ is a UNIX System Services environment variable that allows the mapping of the system time (stored in GMT) to a local time zone. For full details on TZ, see *z/OS V1R8.0 UNIX System Services Command Reference*, SA22-7802.

We set TZ=EST5EDT.

For a full description of all the CICS TG environment variables, see *CICS Transaction Gateway on z/OS Administration*, SC34-6672.

Order of precedence of CICS TG environment variables

Environment variables passed by CTGBATCH to the Gateway daemon can be set in one of the following ways:

- Using the **ctgenvar** script file
- Within the JCL using a data set referenced by the STDENV DD card

The **ctgenvar** script is best used when you run the Gateway daemon from the OMVS command line using the **ctgstart** command (Figure 2-7 on page 45).

The search order for environment variables used by CTGBATCH is as follows:

1. If the STDENV DD statement is defined in the startup JCL, environment variables specified in the reference file are used.
2. If the CTGENVVAR environment variable is set, the variables are taken from the referenced file and these override environment variables set by STDENV.
3. If CTGENVVAR is not set, or the ctgenvar file, which the CTGENVVAR variable specifies does not exist, the Gateway daemon looks in the <install_path>/bin directory for a ctgenvar file. If it finds one, any environment variables that it contains override those already set by the STDENV referenced file.

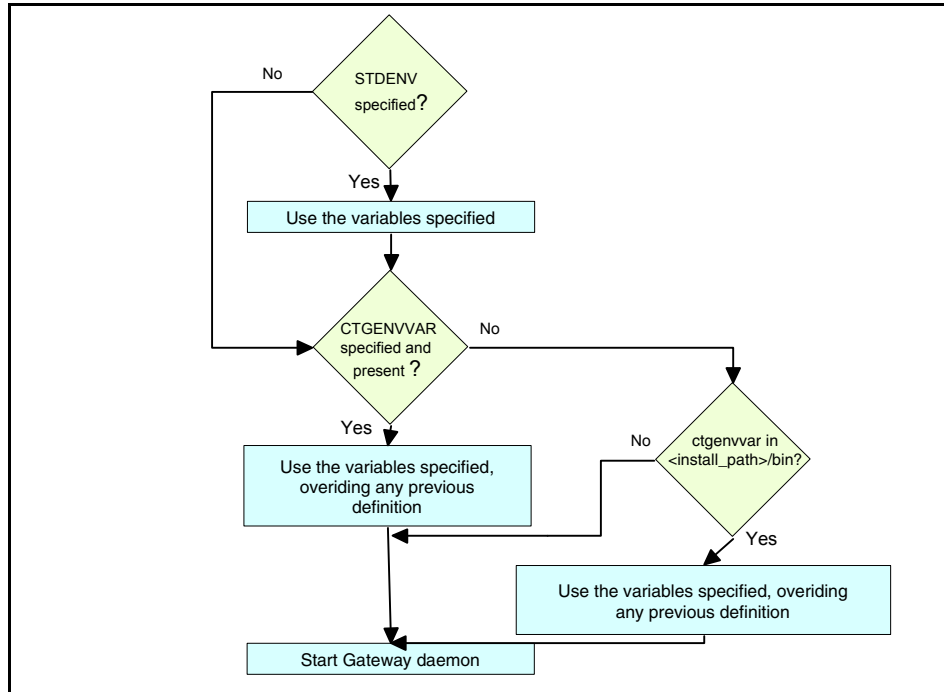


Figure 2-7 Concatenation sequence for CICS TG variables

For ease of maintenance, we placed our environment variables in an STDENV member.

2.3.5 Creating the started task JCL

In 2.6.1, “Starting the Gateway daemon” on page 62, we discuss the different ways in which the Gateway daemon can be started. We recommend that you start the Gateway daemon with a started task job. A sample is supplied in member CTGPROC of SCTGSAMP that you can use as a base for the Gateway daemon started task JCL.

In our scenario, we copied CTGPROC, removed the comments, and specified values for CTGUSR, CTGHLQ, REGION, and the ctgstart path, as shown in Example 2-6.

Example 2-6 JCL for Gateway daemon started task

```
//SCSCT711 PROC
//* CTG V7 start proc
// SET CTGHOME='/usr/lpp/cicstg/ctg710'
// SET CTGHLQ='CTG.V7R1M0'
// SET CTGUSR='CICSSYSF.CTG71.STDENV(SCSCT711)'
// SET LEOPTS=''
//CTG      EXEC PGM=CTGBATCH,REGION=500M,
// PARM='&LEOPTS.&CTGHOME./bin/ctgstart'
//STEPLIB DD DSN=&CTGHLQ..SCTGLOAD,DISP=SHR
//          DD DSN=CICSTS32.CICS.SDFHEXCI,DISP=SHR
//CTGDBG   DD DUMMY
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//STDENV   DD DSN=&CTGUSR.,DISP=SHR
//
```

We recommend that you specify a specific REGION size and do not specify REGION=0M. While REGION=0M requests that the job receives as large a region as it requires, the actual allocation can be limited by the z/OS IEFUSI exit, resulting in a region size that is much smaller than expected.

The IEFUSI exit is used in a z/OS system to limit the REGION size that is allowed to jobs and can override the allocation with a much lower system default value. This limitation can result in a Gateway daemon running with an unexpectedly small storage allocation, suffering storage and performance problems.

If a specific value is set for REGION and the requested storage is not available, the job will fail, allowing early problem determination. We specified a REGION size of 500M, due to the memory required for the number of connection manager threads (500).

2.3.6 Configuring CICS TG for RRMS

We intend to monitor CICS TG transactional (XA) support using workload balancing in our tests. Therefore, we needed to configure CICS TG for Resource Recovery Management Services (RRMS). This section explains how we did that. RRMS support is coordinated for a CICS TG group by a single CICS Transaction Gateway master process called *ctgmaster* located in <install_path>/bin/ctgmaster. Figure 2-8 on page 49 shows the master process in relation to our basic configuration.

The first step in configuring CICS TG for RRMS was to define a CICS master process (ctgmaster). In our environment, we ran our CICS TG master process similarly to our Gateway daemons, as started tasks using CTGBATCH (see Example 2-7). A sample job for ctgmaster can be found in the SDFHSAMP dataset (CTG.V7R1M0.SCTGSAMP(CTGMASTR)).

Example 2-7 Started task JCL for master process SCSCT710

```
//SCSCTG70 PROC
/* CTG Master start proc
/* Created SPK 2/10/7
// SET CTGHOME='/usr/lpp/cicstg/ctg710'
// SET CTGHLQ='CTG.V7R1M0'
// SET CTGUSR='CICSSYSF.CTG71.STDENV(SCSCT710)'
// SET LEOPTS='/'
//MASTER EXEC PGM=CTGBATCH,
//          PARM='&LEOPTS.&CTGHOME./bin/ctgmaster'
//STEPLIB DD DSN=&CTGHLQ..SCTGLOAD,DISP=SHR
//STDOUT DD SYSOUT=*
//STDERR DD SYSOUT=*
//STDENV DD DSN=&CTGUSR.,DISP=SHR
//
```

Configure ctgmaster environment variables

The ctgmaster process has its own environment variables. We defined these in a member in our STDENV library (CICSSYSF.CTG71.STDENV(SCSCT710)): Here is a list of the variables with their description:

- ▶ CTG_MASTER_INPUT
- ▶ CTG_MASTER_NATLANG
- ▶ CTG_MASTER_RRMNAME
- ▶ CTG_MASTER_TRACE_ON
- ▶ BPX_SHAREAS

They are defined as follows:

CTG_MASTER_INPUT

Used to define whether the ctgmaster will listen on the stdin input, or the z/OS console for systems management input. We set CTG_MASTER_INPUT=CONSOLE.

CTG_MASTER_NATLANG

Specifies the language of the message bundle used by the ctgmaster process. We set CTG_MASTER_NATLANG=EN.

CTG_MASTER_RRMNAME

Name of the ctgmaster process managing XA support. This environment variable is used in the Gateway daemon to specify the ctgmaster process. We set CTG_MASTER_RRMNAME=SCSCT710.

CTG_MASTER_TRACE_ON

Set to define a HFS destination for ctgmaster trace data. If undefined, trace data is written to stderr. We set CTG_MASTER_TRACE_ON=NO.

_BPX_SHAREAS=YES

The Gateway daemon can be started in its own address space or share the address space of the process that started it. If you are using CTGBATCH to start the Gateway daemon, ensure that _BPX_SHAREAS=YES is set in the STDENV DD statement.

Configure the Gateway daemon's environment variables

For each Gateway daemon, two environment variables need to be set to register CICS TG for RRMS:

► CTG_RRMNAME

Set with the name with which the Gateway daemon would register with RRS (see 2.3.4, “Creating an STDENV file” on page 40 for more details of this variable). We set this variable to CTG_RRMNAME=SCSCT711.IBM.UA.

► CTG_MASTER_RRMNAME

Was set to the name of the ctgmaster process. We set this variable to CTG_MASTER_RRMNAME=SCSCT710.

Enabling CTGRRMS services

To enable CTGRRMS, the following steps need to be performed:

► APF-authorize SCTGLINK.

Add CTG.V7R1M0.SCTGLINK to the MVS LNKLIST. This load library must be APF-authorized. Then issue the MVS command F LLA,REFRESH to refresh the LNKLIST LOOKASIDE address space (LLA).

► Grant UPDATE authority to the RACF entity CTG.RRMS.SERVICE.

Give the user ID that is running the CICS Transaction Gateway (CTGUSER) UPDATE authority to the RACF entity CTG.RRMS.SERVICE in the FACILITY class. We used the following TSO commands:

– To create the entity in the FACILITY class, run:

RDEFINE FACILITY CTG.RRMS.SERVICE UACC(NONE)

- To enable the user ID for the CICS Transaction Gateway (gway_id), run:
PERMIT CTG.RRMS.SERVICE CLASS(FACILITY) CTGUSER
ACCESS(UPDATE)
- Turn on XA SUPPORT.
Define XASUPPORT in the configuration file /ctg/scsctg71/scstg711/ctg.ini
for each Gateway daemon. We set xasupport=yes.

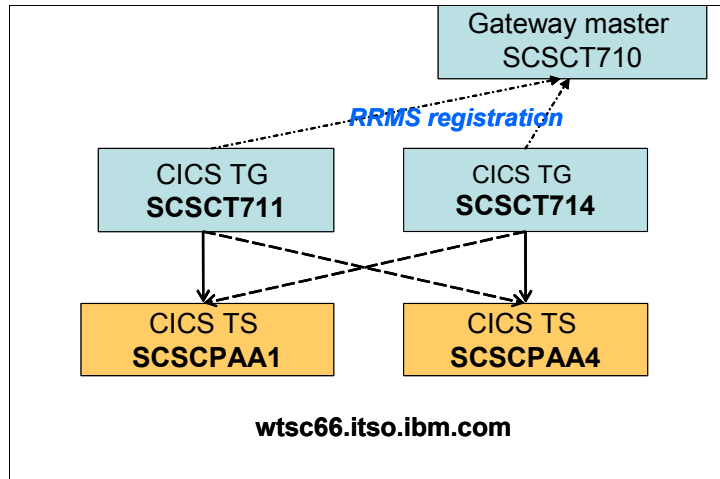


Figure 2-8 Basic configuration with RRMS ctgmaster

2.3.7 Configuring CICS TG for SMF recording

Prior to CICS Transaction Gateway on z/OS V7.1, statistics were captured but could only be viewed after a MVS modify command (see Example 2-8) was issued against the Gateway daemon. The statistics could then be viewed using SDSF to look at the console or JESMSGGLG.

Example 2-8 /F SCSCT711,APPL=STATS,GS=GD:SE used to collect GD and SE stats

```

F SCSCT711,APPL=STATS,GS=GD:SE
BPXM023I (CTGUSER) 038
CTG8239I Response received from CICS Transaction Gateway
SE - System Environment
  SE_SHEAPINIT=134217728 (JVM initial heap size)
  SE_SHEAPMAX=134217728 (JVM maximum heap size)
  SE_CHEAPGCMIN=0 (JVM heap size after GC)
  SE_IGCTIME=0 (JVM GC time)
  SE_LGCTIME=0 (JVM GC time)
  SE_IGCCOUNT=0 (JVM GC count)
  
```

```

SE_LGCCOUNT=0 (JVM GC count)
SE_SELIM=524288000 (Amount of available memory ELIM)
SE_CELOAL=434024448 (Amount of used memory ELOAL)
GD - Gateway daemon
GD_CSTATUS=RUNNING (Gateway daemon status)
GD_SVER=7.1.0.0 (CICS TG version)
GD_SAPPLID=          (CICS TG APPLID)
GD_SAPPLIDQ=         (CICS TG APPLID qualifier)
GD_LALLREQ=0 (Number of requests processed)
GD_IALLREQ=0 (Number of requests processed)
GD_LXATXNC=0 (XA commit requests successfully processed)
GD_IXATXNC=0 (XA commit requests successfully processed)
GD_LXATXNR=0 (XA rollback requests successfully processed)
GD_IXATXNR=0 (XA rollback requests successfully processed)
GD_IXATXNR=0 (XA rollback requests successfully processed)
GD_LLWUTXNC=0 (Extended LUW transactions committed)
GD_ILWUTXNC=0 (Extended LUW transactions committed)
GD_LLWUTXNR=0 (Extended LUW transactions rolled back)
GD_ILWUTXNR=0 (Extended LUW transactions rolled back)
GD_LSYNCTXN=0 (Successful SYNCONRETURN transactions)
GD_ISYNCTXN=0 (Successful SYNCONRETURN transactions)
GD_CSYNCTXN=0 (Number of SYNCONRETURN transactions)
GD_CLWUTXN=0 (Number of Extended LUW transactions)
GD_CXATXN=0 (Number of XA transactions)
GD_LXAREQ=0 (Number of XA requests)
GD_IXAREQ=0 (Number of XA requests)
GD_LREQDATA=0 (Amount of Java client request data)
GD_IREQDATA=0 (Amount of Java client request data)
GD_LRESPDATA=0 (Amount of Java client response data)
GD_IRESPDATA=0 (Amount of Java client response data)
GD_LAVRESP=0 (Average Gateway daemon response time)
GD_IAVRESP=0 (Average Gateway daemon response time)
GD_SNAME=SCSCT711 (Gateway daemon name)
GD_CHEALTH=100 (Gateway daemon health)
GD_LRUNTIME=1513 (Gateway daemon running time)
GD_SSTATINT=010000 (Length of the statistics interval HHMMSS)
GD_SSTATEOD=000000 (Logical End Of Day time HHMMSS)
GD_CNEXTRESET=060000 (End of interval time HHMMSS)
GD_IRUNTIME=1514 (Interval running time)

```

CICS Transaction Gateway on z/OS V7.1 supports statistics being written to System Management Facility (SMF). CICS TG writes Interval and End-of-day statistics to SMF as type 111 records. For further details on the statistics collected, see *CICS TG z/OS Administration*, SC34-6754.

To configure CICS TG to collect SMF statistics, the following steps need to be performed:

- ▶ Activate SMF recording in the configuration file.
- ▶ Grant the Gateway daemon user ID read access to RACF facility BPX.SMP.
- ▶ Verify CICS TG statistic records are being captured.

Activating SMF recording

System Management Facility (SMF) can be activated by adding the parameter `statsrecording=on` in the GATEWAY section of the configuration file `ctg.ini`. Interval stats will be collected at a specified interval provided by the `statint=010000` parm in the configuration file `ctg.ini` (see Example 2-9). Configuration file parm `stateod` specifies the time when stats are collected and reset. It is also used to align the time when interval statistics are collected rather than when the Gateway daemon is started. For example, in our case, we allowed `stateod` to default to `000000` (midnight) and the interval period was set to `010000` (one hour) so interval times would be on the hour.

Example 2-9 ctg.ini file showing SMF parms

```
BROWSE -- /ctg/scsctg71/scstg714/ctg.ini ----- Line 00000015 Col
...
healthreporting=on
statsrecording=on
statint=010000
protocol@tcp.handler=com.ibm.ctg.server.TCPHandler
```

Granting READ access to the BPX.SMF facility

To write to SMF, the user ID of the CICS Gateway daemon must be permitted READ access to the BPX.SMF facility. Example 2-10 shows the command used to grant CTGUSER access to this RACF facility.

Example 2-10 RACF command to permit CTGUSER access to BPX.SMF facility

```
PERMIT BPX.SMF CLASS(FACILITY) ACCESS(READ) ID(CTGUSER)
```

Verify CICS TG is writing statistic records to SMF

To check whether SMF records were being written, we extracted the CICS TG SMF type 111 records from the z/OS SMF datasets and ran the statistic reporter. We explain how to do this and show the output produced in 8.2.2, “CICS TG SMF recording” on page 295.

2.4 Configuring CICS TS

In this section, we deal with the following CICS TS configuration tasks:

1. Creating a CONNECTION definition.
2. Creating a SESSIONS definition.
3. Installing the definitions.
4. Configuring CICS for RRMS.
5. Compiling and installing the sample program.
6. Editing and assembling DFHCNV.
7. Opening interregion communication.

2.4.1 Creating a CONNECTION definition

An EXCI connection for the CICS TG must be defined in CICS. You can copy the supplied EXCS CONNECTION definition from the DFH\$EXCI group and update it to reflect the required definitions.

You can install either *generic* or *specific* EXCI connections in your CICS regions to allow them to communicate with the Gateway daemon. For a specific EXCI connection, the value specified to the Gateway daemon environment variable DFHJVPIPE must be the same as the NETNAME option on the connection definition. The NETNAME itself is an arbitrary value. A generic connection is defined in much the same way, except that the NETNAME option is left blank.

Note: We recommend that you use specific EXCI connections because this gives you more control over which Gateway daemons can access which CICS regions and also makes problem determination easier.

We copied the EXCS connection to our own SCST711 group as T711 and set NETNAME to SCST711 (Figure 2-9 on page 53).


```

OBJECT CHARACTERISTICS                                CICS RELEASE = 0650
CEDA View CONNecTion( T711 )
  CONNecTion      : T711
  Group          : SCST711
  Description     : FOR CTG711
CONNECTION IDENTIFIERS
  Netname        : SCST711
  INDSys         :
REMOTE ATTRIBUTES
  REMOTESYSem    :
  REMOTEName     :
  REMOTESYSNet   :
CONNECTION PROPERTIES
  ACcessmethod   : IRc          Vtam | IRc | INdirect | Xm
  PRotocol       : Exci        Appc | Lu61 | Exci
  Conntype       : Specific     Generic | Specific
  SInglesess     : No          No | Yes
  DATAstream    : User        User | 3270 | SCs | STrfield | Lms
+ RECOrdformat   : U          U | Vb

                                SYSID=PAA1 APPLID=SCSCPAA1

PF 1 HELP 2 COM 3 END          6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL

```

Figure 2-9 T711 connection definition

2.4.2 Creating a SESSIONS definition

The EXCS SESSIONS definition can also be copied from group DFH\$EXCI and the connection attribute of the session definition altered to refer to the previously defined CONNECTION. It is also possible to set the RECEIVECOUNT and RECEIVEPFX parameters. It is desirable to do this because:

- ▶ RECEIVECOUNT defines the maximum number of EXCI pipes to be used for the defined connection.
- ▶ RECEIVEPFX allows the definition of a single or double character prefix so that the usage of each session can be easily identified. If a single character RECEIVEPFX is specified, the RECEIVECOUNT can be a maximum of 999. If a two character RECEIVEPFX is specified, the RECEIVECOUNT can be a maximum of 99.

We defined session T711, setting the RECEIVEPFX to 1 and the RECEIVEDCOUNT to 200 (Figure 2-10). See *CICS Transaction Server on z/OS V3.2 CICS Resource Definition Guide* SC34-6815 for more details about the definition of EXCI CONNECTION and SESSIONS.

```

OVERTYPE TO MODIFY                                CICS RELEASE = 0650
CEDA ALTER Sessions( T711      )
Sessions      : T711
Group         : SC SCT711
Description   ==> FOR CTG 711
SESSION IDENTIFIERS
Connection    ==> T711
SESSName      ==>
NETnameq      ==>
MODename      ==>
SESSION PROPERTIES
Protocol      ==> Exci          Appc | Lu61 | Exci
Maximum       ==> 000 , 000     0-999
RECEIVEPfx    ==> 1
RECEIVEDCount ==> 200          1-999
SENDPfx       ==>
SENDCount     ==>              1-999
SENDSize      ==> 04096        1-30720
+ RECEIVESize ==> 04096        1-30720

                                SYSID=PAA1 APPLID=SCSCPAA1

PF 1 HELP 2 COM 3 END          6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL

```

Figure 2-10 T711 SESSIONS definition

2.4.3 Installing the definitions

We activated the CICS definitions by installing the SC SCT711 group using the CEDA INSTALL GROUP(SC SCT711) command (Figure 2-11 on page 55).

After the group is installed, CEMT can be used to view the newly created connection and session definitions (Figure 2-12 on page 55).

```

I CONN
STATUS: RESULTS - OVERTYPE TO MODIFY
Con(T711) Net(SCSCT711)    Ins    Irc Exci
Con(T714) Net(SCSCT714)    Ins    Irc Exci

                                SYSID=PAA1 APPLID=SCSCPAA1
RESPONSE: NORMAL                TIME: 06.24.50 DATE: 10.04.07
PF 1 HELP          3 END          5 VAR          7 SBH 8 SFH 9 MSG 10 SB 11 SF

```

Figure 2-11 CEMT display showing CONN definitions

```

I TERM(1*
STATUS: RESULTS - OVERTYPE TO MODIFY
Ter(11 )      Pri( 000 ) Aut Ins Ati Tti Ses
Net(SCSCT711)                                Rem(T711)
Ter(110 )     Pri( 000 ) Aut Ins Ati Tti Ses
Net(SCSCT711)      Tas(0029984) Rem(T711)
Ter(1100 )    Pri( 000 ) Aut Ins Ati Tti Ses
Net(SCSCT711)                                Rem(T711)
Ter(111 )     Pri( 000 ) Aut Ins Ati Tti Ses
Net(SCSCT711)      Tas(0029985) Rem(T711)
Ter(112 )     Pri( 000 ) Aut Ins Ati Tti Ses
Net(SCSCT711)      Tas(0029990) Rem(T711)
Ter(113 )     Pri( 000 ) Aut Ins Ati Tti Ses
Net(SCSCT711)      Tas(0029992) Rem(T711)
Ter(114 )     Pri( 000 ) Aut Ins Ati Tti Ses
Net(SCSCT711)      Tas(0030013) Rem(T711)
Ter(115 )     Pri( 000 ) Aut Ins Ati Tti Ses
Net(SCSCT711)      Tas(0030014) Rem(T711)
+ Ter(116 )   Pri( 000 ) Aut Ins Ati Tti Ses
Net(SCSCT711)      Tas(0030015) Rem(T711)
E Right parenthesis assumed at end of command.

                                SYSID=PAA1 APPLID=SCSCPAA1
RESPONSE: NORMAL                TIME: 06.31.22 DATE: 10.04.07
PF 1 HELP          3 END          5 VAR          7 SBH 8 SFH 9 MSG 10 SB 11 SF

```

Figure 2-12 CEMT display showing Sessions to SCSCT711

2.4.4 Configuring CICS TS for RRMS

We intend to test CICS TG transactional support in our scenario. Therefore, we configured CICS to register with Resource Recovery Management Services (RRMS).

It is a requirement that RRMS support is functioning and CICS is registered with RRMS if CICS is to handle extended logical units of work or XA support, because RRMS is the resource manager for these extended transactions. In our environment, we enabled CICS registration with RRMS by setting the CICS system initialization (SIT) parameter RRMS=YES as an override.

Tip: It is possible to verify that RRMS is open using the CEMT INQUIRE RRMS command.

2.4.5 Compiling and installing the sample programs

The CICS COBOL sample program ec01.ccp is provided with the CICS TG in the /usr/lpp/cicstg/ctg710/samples/server directory.

For our testing, we copied the program to our CICS source library, compiled the program, and added the load module to our program load library.

2.4.6 Editing and assembling DFHCNV

The test program used in our tests was a Java program that runs in the UNIX System Services environment, which is a UNICODE environment. The Java program receives data that is produced by a COBOL program running in CICS, which is an EBCDIC environment. The data is passed in a COMMAREA. To ensure that this COMMAREA is converted correctly between UNICODE and EBCDIC, we created a DFHCNV table entry for program ECIPROG with SRVERCP=037, CLINTCP=850. For further details about using DFHCNV, refer to the *CICS Transaction Server on z/OS V3.2 Internet Guide*, SC34-6831.

2.4.7 Opening interregion communication

It is possible to check the status of CICS interregion communication using the CEMT INQUIRE IRC command (Figure 2-13 on page 57). If IRC is closed, you can issue the CEMT SET IRC OPEN command to open CICS IRC.

```

I IRC
STATUS: RESULTS - OVERTYPE TO MODIFY
Irc Ope Xcf(DFHIR000)

RESPONSE: NORMAL
PF 1 HELP      3 END      5 VAR      7 SBH 8 SFH 9 MSG 10 SB 11 SF

SYSID=PAA1 APPLID=SCSCPAA4
TIME: 12.07.40 DATE: 10.03.07

```

Figure 2-13 CEMT INQUIRE IRC command

2.5 Testing the configuration

This section explains how to verify that your Gateway daemon is functioning correctly using:

- ▶ The CTGTESTR sample JCL
- ▶ CICS TG STDOUT output
- ▶ The **netstat** command
- ▶ The **ping** command
- ▶ The **ctgping** utility

2.5.1 CTGTESTR sample program

To test our configuration, we used the CTGTESTR sample job supplied in the SCTGSAMP data set. This job runs the **ctgtest** script from the <install_path>/samples directory. The **ctgtest** script is designed to be run without user input so that it can be executed from a batch job. The script uses the EciB1 sample program.

We updated the CTGTESTR job (Figure 2-12 on page 55) with the following:

- ▶ Host name and port used by our Gateway daemon
- ▶ PATH variable set to the HFS directory of the Java binary directory
- ▶ CLASSPATH variable set to the HFS directory of ctgclient.jar

Note: Information about the CTGTESTR sample, including parms syntax, can be found in /usr/lpp/cicstg/ctg710/samples/samples.txt.

Example 2-11 CTGTESTR job control

```
/ SET CTGHLQ='CTG.V7R1M0'  
// SET LEOPTS='/'  
// SET USR='/usr/lpp/cicstg/ctg710/samples'  
//TEST      EXEC  PGM=CTGBATCH,REGION=250M,  
// PARM='%LEOPTS.&USR./ctgtest CTGUSER PASS wtsc66.itso.ibm.com 2006'  
//STEPLIB DD DSN=&CTGHLQ..SCTGLOAD,DISP=SHR  
//STDOUT   DD    SYSOUT=*  
//STDERR   DD    SYSOUT=*  
//STDENV   DD    *  
PATH=/bin:/usr/lpp/java/J5.0/bin  
CLASSPATH=/usr/lpp/cicstg/ctg710/classes/ctgclient.jar  
/*  
//
```

EciB1 flows an ECI request to a connected CICS region through a Gateway daemon (Figure 2-14 on page 59).

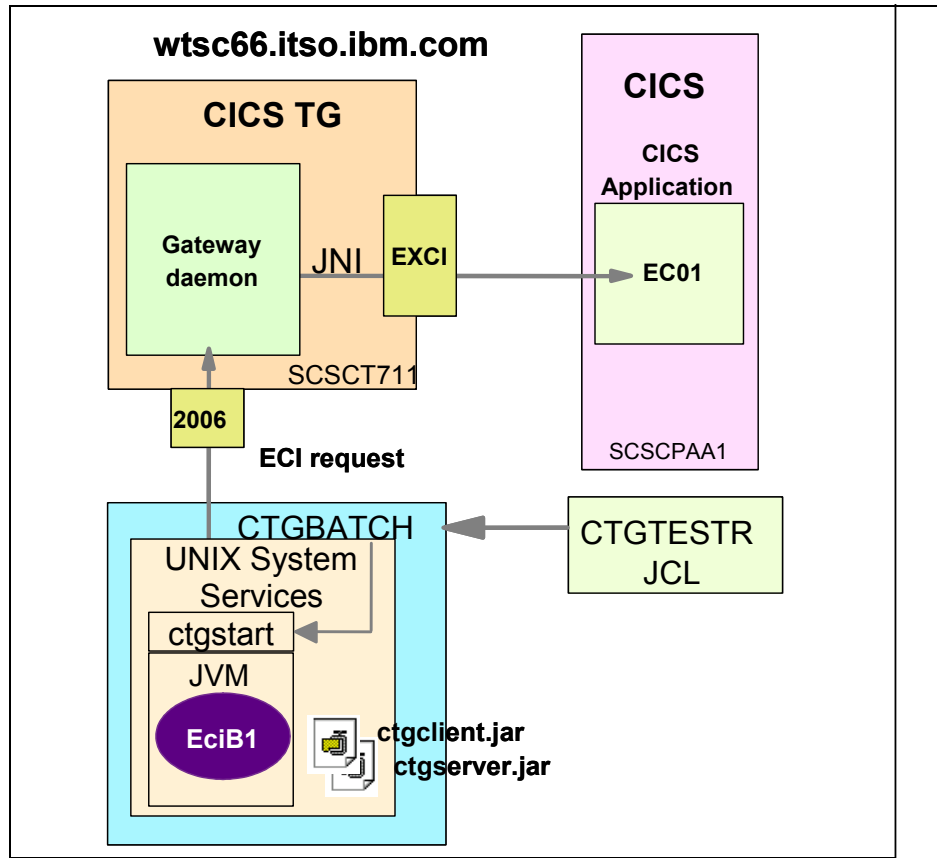


Figure 2-14 CTGTESTR test scenario

The output of this application is the date and time as formatted by the CICS COBOL program EC01.

2.5.2 Checking the Gateway daemon STDOUT

When the Gateway daemon started (see 2.6.1, “Starting the Gateway daemon” on page 62), we checked the STDOUT to ensure that the Gateway daemon had started correctly (Example 2-12). Message CTG6524I shows that the TCP protocol handler has started successfully so that the Gateway daemon can accept incoming work. Message CTG5612I shows that the Gateway daemon has successfully initialized.

Example 2-12 Successful start of Gateway daemon as seen in STDOUT

```
CTG8400I Using configuration file /ctg/scsctg71/scstg711/ctg.ini.
CTG6577I Java version is 1.5.0
CTG6737I XA transaction support is enabled
CTG8421I SMF Recording is enabled
CTG9214I The Gateway daemon is running in a TCP/IP load balancing group with
resource manager name 'SCSCT710'.
CTG9223I Successfully attached to the ctgmaster RRMS registration.
CTG9296I The Gateway daemon is using resource manager name 'SCSCT711.IBM.UA' to
support extended LUW based transactions.
CTG8928I User ID and password authentication is not enabled.
((AUTH_USERID_PASSWORD=NO).
CTG6898I 100 EXCI pipes are available for use by the CICS Transaction Gateway.
CTG6981I Successfully initialized JNI library
CTG6422I Health reporting is enabled.
CTG6502I Initial ConnectionManagers = 100, Maximum ConnectionManagers = 100
CTG6526I Initial Workers = 100, Maximum Workers = 100
CTG6505I Successfully created the initial ConnectionManager and Worker threads.
CTG6524I Successfully started handler for the tcp: protocol on port 2006
CTG6524I Successfully started handler for the statsapi: protocol on port 2981
CTG6512I CICS Transaction Gateway initialization complete
```

2.5.3 TCP/IP netstat command

We checked the state of the TCP/IP sockets in use by our Gateway daemon using the TSO command NETSTAT ALLC (CLI SCSCT711). Example 2-13 shows the output of this command with the Gateway daemon listening on port 2006 for TCP connections and 2981 for the statistics interface in use by OMEGAMON XE.

Example 2-13 TSO NETSTAT ALLC (CLI SCSCT711)

MVS	TCP/IP	NETSTAT	CS	V1R8	TCPIP Name:	TCPIP	08:49:24
User Id	Conn	Local	Socket		Foreign	Socket	State
-----	----	-----			-----		-----
SCSCT711	000C7BC8	0.0.0.0..2007			0.0.0.0..0		Listen
SCSCT711	000C7BC9	127.0.0.1..2981			0.0.0.0..0		Listen

```
SCSCT711 000C7BD3 127.0.0.1..2981      127.0.0.1..2835      Establish
***
```

2.5.4 Using the ping command

We checked basic IP connectivity from our workstation to z/OS using the **ping** command from a Windows 2000 command prompt (Example 2-14).

Example 2-14 Ping to z/OS verifying hosts or DNS setup

```
C:\>ping wtsc66.itso.ibm.com

Pinging wtsc66.itso.ibm.com [9.12.4.75] with 32 bytes of data:

Reply from 9.12.4.75: bytes=32 time=125ms TTL=50
Reply from 9.12.4.75: bytes=32 time=109ms TTL=50
Reply from 9.12.4.75: bytes=32 time=109ms TTL=50
Reply from 9.12.4.75: bytes=32 time=109ms TTL=50

Ping statistics for 9.12.4.75:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 109ms, Maximum = 125ms, Average = 113ms
```

2.5.5 Running CTGTESTR

We performed our basic test by submitting the CTGTESTR job. The job finished with a completion code of 0. Checking the job output, we could see the date and time returned by the CICS program EC01 (Example 2-15).

Example 2-15 CTGTESTR test output

```
ctgtest: Executing EciB1 with userid = CTGUSER and parameters:
wtsc66.itso.ibm.com 2006
CICS Transaction Gateway Basic ECI Sample 1
Usage: java com.ibm.ctg.samples.eci.EciB1 Gateway URL
                                         Gateway Port Number
                                         SSL Keyring
                                         SSL Password

To enable client tracing, run the sample with the following Java option:
-Dgateway.T.trace=on
The address of the Gateway has been set to wtsc66.itso.ibm.com Port:2006
CICS Servers Defined:
  1. SCSCPAA4 -CICS 3.2
  2. SCSCPAA1 -CICS 3.2
Choose Server to connect to, or q to quit:
Program EC01 returned with data:-
```

Hex: 30342f31302f30372031303a34343a34340
ASCII text: 04/10/07 10:44:44

Note that the completion code indicates whether the CTGTESTR job successfully completed or not, rather than the result of the ECI call itself. The completion code is still 0 even if the CICS region is not active; however, the job output shows an ECI_ERR_NO_CICS error (Example).

```
ctgtest: Executing EciB1 with userid = CTGUSER and parameters:
wtsc66.itso.ibm.com 2006
CICS Transaction Gateway Basic ECI Sample 1
Usage: java com.ibm.ctg.samples.eci.EciB1 Gateway URL
                                     Gateway Port Number
                                     SSL Keyring
                                     SSL Password To enable
client tracing, run the sample with the following Java option:
-Dgateway.T.trace=on
The address of the Gateway has been set to wtsc66.itso.ibm.com
Port:2006
CICS Servers Defined:
  1. SCSCPAA4 -CICS 3.2
  2. SCSCPAA1 -CICS 3.2
Choose Server to connect to, or q to quit:
ECI returned: ECI_ERR_NO_CICS
Abend code was
```

Figure 2-15 Results of CTGTEST when CICS is not available

2.6 Operating CICS TG

In this section, we discuss the operation of CICS TG, including how to do the following:

1. Starting the Gateway daemon.
2. Stopping the Gateway daemon.

2.6.1 Starting the Gateway daemon

You can start the Gateway daemon either from a UNIX System Services command line or by a JCL batch job. We do not discuss the use of the **ctgstart** script from UNIX System Services because the use of a batch job has many advantages, including more flexibility in the routing of Gateway daemon messages and the option to use ARM.

There are two alternatives for starting the Gateway daemon from JCL:

- ▶ BPXBATCH
- ▶ CTGBATCH

BPXBATCH

Prior to Version 6 of CICS TG, starting the Gateway daemon was dependent on the z/OS supplied utility BPXBATCH.

Starting the Gateway daemon using BPXBATCH is still supported with CICS TG V7.1 and is described in *CICS Transaction Gateway on z/OS Administration*, SC34-6672. However, the recommended way to start the Gateway daemon from JCL is to use CTGBATCH.

CTGBATCH

CTGBATCH is a Language Environment batch mode program that is supplied with CICS TG V6 and later. It is used to start the Gateway daemon through the JCL started task and to pass the required environment variables to the UNIX System Services **ctgstart** script.

CTGBATCH has advantages over BPXBATCH in that it can route Gateway daemon messages to the following destinations:

- ▶ The JES log
- ▶ An MVS sequential data set
- ▶ An HFS file

If you start the Gateway daemon from JCL using CTGBATCH (or BPXBATCH), it can register with the z/OS Automatic Restart Manager component. You can start CTGBATCH from a JCL batch job or as a started task procedure. We recommend that you start the Gateway daemon through CTGBATCH from a started task procedure.

We started the Gateway daemon using a procedure called SCSC711 in ESA.SYS1.PROCLIB (see Example 2-6 on page 46). The procedure is started as a started task from SDSF using the /S SCSC711 command, as shown in Example 2-16.

Example 2-16 Results of /S SCSC711 written to the SDSF log

```
S SCSC711
$HASP100 SCSC711 ON STCINRDR
IEF695I START SCSC711 WITH JOBNAME SCSC711 IS ASSIGNED TO USER
CTGUSER , GROUP CICS
$HASP373 SCSC711 STARTED
```

2.6.2 Stopping the Gateway daemon

In CICS TG V7.1, you can stop the Gateway daemon in a controlled manner, rather than using the SDSF CANCEL command.

Normal shutdown

A normal shutdown quiesces the Gateway daemon so that all the work that is in progress completes but no new work is started. After all the work in progress has completed, the Gateway daemon then shuts down.

A normal shutdown is requested with the following MVS command:

```
/F JOB_NAME,APPL=SHUT|SHUTDOWN
```

Tip: The MVS command /P JOB_NAME that specifies the Gateway daemon job name has exactly the same effect as /F JOB_NAME,APPL=SHUT (as shown in Example 2-17).

Example 2-17 Normal shutdown using /P SCSC711

```
P SCSC711
BPXM023I (CTGUSER) 134
CTG8239I Response received from CICS Transaction Gateway
CTG8235I Normal shutdown was requested
```

Immediate shutdown

An immediate shutdown terminates all work in progress and breaks any active connections. The Gateway daemon does not accept any new connections and shuts down immediately.

An immediate shutdown is requested with the following MVS command, as shown in Example 2-18:

```
/F JOB_NAME,APPL=SHUT|SHUTDOWN,IMM|IMMEDIATE
```

Example 2-18 Immediate shutdown using /F SCSC711,APPL=SHUT,IMM

```
F SCSC711,APPL=SHUT,IMM
BPXM023I (CTGUSER) 336
CTG8239I Response received from CICS Transaction Gateway
CTG8234I Immediate shutdown was requested
BPXM023I (CTGUSER) CTG6509I SCSC711 IMMEDIATE SHUTDOWN OF GATEWAY DAEMON
STARTED BY Z/OS OPERATOR
BPXM023I (CTGUSER) CTG6511I SCSC711 GATEWAY DAEMON HAS SHUT DOWN
```

Note: If you have requested a normal shutdown and the Gateway daemon is taking too long to shut down, it is possible to request an immediate shutdown subsequently.

2.6.3 Configuring for multiple Gateway daemons

After we had tested that our single Gateway daemon could communicate with a CICS TS region successfully, we needed to add a second Gateway daemon to our base configuration. See Table 2-3 for the definitions used when adding the second CICS TG and associated CICS TS.

Table 2-3 Definitions checklist with second Gateway daemon and CICS regions

Value	Gateway daemon	Second Gateway daemon	CICS TS Region 1	CICS TS Region2
Host name	wtsc66.itso.ibm.com	wtsc66.itso.ibm.com	-	-
IP address	9.12.4.75	9.12.4.75	-	-
TCP/IP port	2006 ^a	2006	-	-
Statsport	2981	2984	-	-
Job name	SCSCT711	SCSCT714	SCSCPAA1	SCSCPAA4
APPLID	SCSCT711	SCSCT711	SCSCPAA1	SCSCPAA4
NETNAME	SCSCT711	SCSCT714	-	-
CONNECTION	-	-	T711 + T714	T711 + T714

a. Both Gateway daemons have the same port number 2006 because we are implementing TCP/IP port sharing (see “Implementing TCP/IP port sharing” on page 67).

The following steps need to be performed when setting up the second Gateway daemon:

1. Set RACF permissions on the Gateway daemon started task user ID.
2. Create started task JCL.
3. Create new directories in UNIX System Services.
4. Create a new configuration file.
5. Create a new STDENV member.
6. Create CICS definitions.

The following list contains what we did in each step in this process. However, because we assume that you have performed these tasks previously to set up the first Gateway daemon, we do not go into each step in detail.

To set up a second Gateway daemon, do the following:

1. Set RACF permissions on the Gateway daemon started task user ID.

We used the same user ID (CTGUSER) for each gateway. This user ID has already got permission to start tasks with prefix SCST71, so we did not need to add any further permission.

2. Create started task JCL in the PROCLIB.

We copied our SCST711 started task job as SCST714 and changed it to reference a new STDENV member in SYSFCICS.CTG71.STDENV(SCST714).

3. Create new directories in UNIX System Services.

We created a HFS structure for this Gateway daemon (see 2.3.1, “Defining an HFS” on page 33) with directories for logs and tmp directories mounted on separate MVS datasets. We then gave the Gateway daemon the necessary permissions to access the directories (as shown in 2.3.2, “Setting directory permissions” on page 37).

4. Create a new configuration file.

We copied /ctg/ctg710/scstg711/ctg.ini to /ctg/scstg71/scstg714/ctg.ini. The ‘statsport’ and ‘tfile’ settings were changed for SCST714.

5. Create a new STDENV member.

We copied CICSSYSF.CTG71.STDENV(SCST711) to CICSSYSF.CTG71.STDENV(SCST714) and made the following changes:

- CICSCLI to /ctg/scstg71/scstg714/ctg.ini
- CTG_JNI_TRACE to /ctg/scstg71/scstg714/logs/jni.trace
- _CEE_DMPTARG to /ctg/scstg71/scstg714/logs
- DFHJVPIPE to SCST714 to match the NETNAME which we specified on our CONNECTION definition in CICS region SCSCPAA4
- DFHJVSYSTEM_00 to SCSCPAA4 and DFHJVSYSTEM_01 to SCSCPAA1
- CTG_RRMNAME to SCST714.IBM.UA
- CTG_MASTER_RRMNAME to SCST710
- TMPDIR to /ctg/scstg71/scstg714/tmp

6. Create CICS definitions.

In both CICS TS regions (SCSCPAA1 and SCSCPAA4), we defined the following resources:

- **CONNECTION:** We created a T714 CONNECTION definition and set NETNAME to SCST714.
- **SESSIONS:** We created a T714 SESSION definition and set it the CONNECTION to reference T714 and the RECEIVEPFX to 4.

Implementing TCP/IP port sharing

CICS TG on z/OS supports TCP/IP load balancing. When TCP/IP port sharing is used, requests for work to be shared between several Gateway daemons are received through a single TCP/IP port. This means that the Java client application will send a request to a single port and TCP/IP will establish a connection with one of several Gateway daemons listening on the same port number. Once established, the Java application will continue to use the same gateway daemon connection.

We decided to use this feature of TCP/IP to allow connection balancing across our Gateway daemons to provide high availability by distributing the connections across two Gateways. This would enable failover in the case of an individual component failure.

Figure 2-16 shows our configuration with port sharing included.

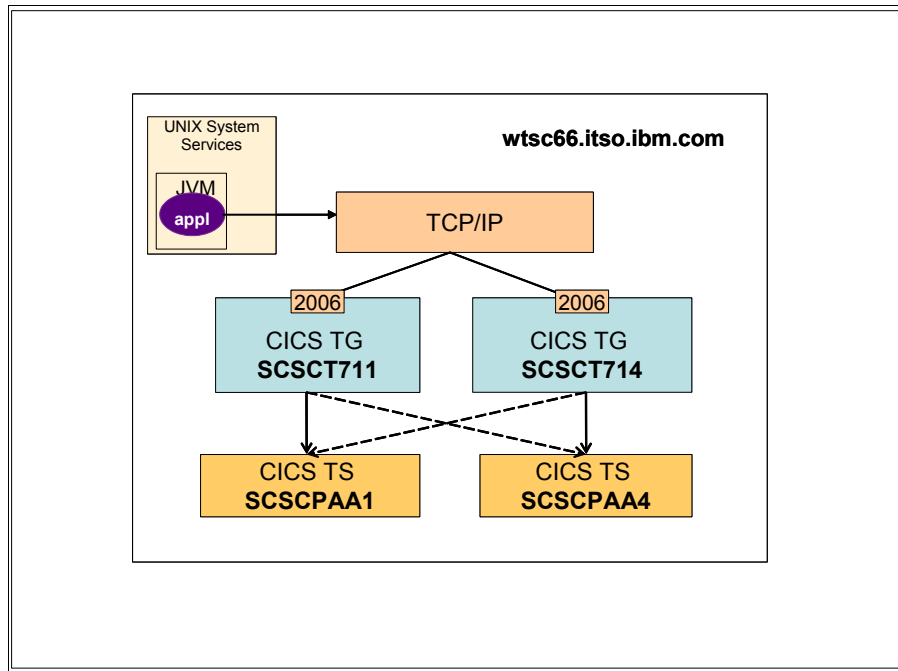


Figure 2-16 TCP/IP port sharing across SCSC711 and SCSC714 on port 2006

To implement port sharing on our LPAR:

- ▶ TCP/IP parm member TCP PROF in TCP.SC66.TCPPARMS was changed. We included entries, in the TCP/IP parms dataset, for our Gateway daemons (SCSC711 and SCSC714) (see Example 2-19).
- ▶ In the Gateway daemon configuration file (ctg.ini), we set the port number in the protocol@tcp.parameters=port statement to the same value(2006), for each of the Gateway daemons sharing the port.
- ▶ Then we brought online the new TCP parms using the following MVS command:


```
V TCPIP,TCP,0,DSN=TCP.SC66.TCPPARMS(TCP PROF)
```

Example 2-19 Entries in TCPPARMS member to enable port sharing

```

BROWSE    TCP.SC66.TCPPARMS(TCP PROF) - 01.51          Line 00000109 Col 001 080
Command ==>                                         Scroll ==> CSR
3335 TCP EZAIMSJL          ; IMS Listener BMP
5001 TCP TPS              ; OS/390 TME 10 GEM Connection Service
5314 TCP SCSCPTA1 SHAREPORT ;
5314 TCP SCSCPTA2 SHAREPORT ;

```


2006 TCP SCSC711 SHAREPORTWLM; CICS TG

2006 TCP SCSC714 SHAREPORTWLM; CICS TG

8083 TCP SCSCPLA1 SHAREPORT ; CICS logical EJB server - listener region

SHAREPORT enables a group of servers to listen on the same port, thereby sharing the incoming workload. This algorithm evenly balances the number of active connections across the available server without regard to how well a given server was performing.

SHAREPORTWLM will distribute connections in a weighted round-robin fashion based on the WLM server-specific recommendations for the server. The CICS TG V7.1 can update its WLM server-specific recommendations based on the success rate of calls to CICS servers. For more details on how we exploited this function, refer to “Testing the use of the CICS TG health reporting” on page 268.

2.7 Migrating to CICS TG V7.1

In this section, we discuss:

1. Migrating from `ctgenvar`.
2. Other migration and configuration considerations.

2.7.1 Migrating from `ctgenvar`

The sample `ctgconvnv` script is supplied in the `<install_path>/bin` directory. You can use this script to migrate `ctgenvar` settings from a pre-CICS TG V6 system. The script converts a `ctgenvar` script into the STDENV style settings, which can be used by CTGBATCH. You can invoke the `ctgconvnv` script from:

- ▶ A UNIX System Services command line with the following command:
`ctgconvnv [wrap width] [source file name]`
- ▶ A batch job. A sample job to invoke `ctgconvnv` is supplied in `SCTGSAMP(CTGCONV)`.

Note that `ctgconvnv` is not tolerant of environment variables that it does not recognize. If it finds a variable that it cannot identify, it stops parsing `ctgenvar`.

Tip: If the `ctgenvar` script that is specified in the `ctgconvnv` command does not have the execute permissions set for the user ID running `ctgconvnv`, you will receive the following message:

CTG6124W Source file filename not found or not executable.

2.7.2 Other migration and configuration considerations

In this section, we list other migration and configuration considerations.

Default configuration file name

The default configuration file name has changed from CTG.INI in CICS TG V5 to ctg.ini. If both exist in the <install_path>/bin directory, the ctg.ini file will be used.

You should specify the configuration file in the CICSCLI environment variable to ensure that the Gateway daemon starts with the required settings.

Configuration tool

There are two variations of the CICS TG configuration tool that is shipped with CICS TG V7.1:

- ▶ The **ctgcfg** tool is supplied in the <install_path>/bin directory and runs on z/OS and is accessed through an X Window System interface.
- ▶ The **ctgfzoscfg** tool is supplied in the <install_path>/ctgzoscfg directory as a zipped file. It can be transferred to a distributed platform using FTP.

Both tools generate a configuration file and a **ctgenvvar** script. The advantage of using these tools is that they perform validation checking on the values specified.

We did not use a configuration tool.

Use of AUTH_USERID_PASSWORD environment variable

This variable determines whether RACF is used to authenticate user IDs and passwords. In versions of the CICS Transaction Gateway before Version 7, RACF authentication is enabled if the environment variable is set to any value, including NO. This has now been corrected.

We used AUTH_USERID_PASSWORD=NO, meaning RACF authorization was not enabled.

Features removed

In CICS TG V7.1 the following features are removed:

- ▶ Removal of configuration setting sotimeout
- ▶ Removal of UNIX System Services utility ctgarm

2.8 Problem determination

In this section, we present information that we learned while installing the CICS TG and general information about problem determination and tracing.

2.8.1 Tips and utilities

This section includes useful commands and utilities for debugging problems with your configuration.

z/OS TCP/IP commands

We found the following TCP/IP commands useful when analyzing network problems with our CICS TS:

- ▶ **HOMETEST**

Issue this as a TSO command. If there is a valid TCP/IP address or host name, this command tells you what it is, along with other configuration information.

- ▶ **NETSTAT**

Issue this command to obtain information about the TCP/IP sockets that are in use (TSO and other platforms). The UNIX System Services version of this command can be invoked using the **onetstat** command.

- ▶ **NSLOOKUP <IP address>**

Issue this as a TSO command. A valid TCP/IP address tells you the host name and vice versa.

- ▶ **PING**

Use this to determine if an address is active and to resolve host names to IP addresses (Example 2-20).

Example 2-20 Pinging the z/OS host

```
tso ping wtsc66.itso.ibm.com
CS V1R8: Pinging host WTSC66.ITS0.IBM.COM (9.12.4.75)
Ping #1 response took 0.000 seconds.
***
```

CTGINFO including Java version

To display the level of the Java Development Kit (JDK™) installed on your system, use a MVS modify command /F SCSC711,APPL=DUMP,CTGINFO to display ctginfo, including Java version on STDOUT, as shown in Example 2-21.

Example 2-21 Extract from STDOUT showing ctginfo including Java level

```
10/22/07 07:59:42:443 Y0 CTG8268I Invoking dump request.
10/22/07 07:59:42:449 Y8251 CICS TG Informational Dump:
CICS Transaction Gateway version 7.1.0.0
Build level c000-20070921
Trace started Oct 22, 2007 7:59:42 AM
Class version @(#) java/com/ibm/ctg/client/T.java, cd_gw_logandtrace,
c000 1.25.5.18 07/05/30 15:41:19
System properties :
java.version = 1.5.0
java.vendor = IBM Corporation
java.class.version = 49.0
OS details :- z/OS 01.08.00 s390
bitmode = 32
java.class.path =
/usr/lpp/cicstg/ctg710/classes/ctgserver.jar:/usr/lpp/cicstg/ctg710/c
lasses/ctgclient.jar:/usr/lpp/cicstg/ctg710/c
lasses/ctgsamples.jar
java.library.path =
/Z18RD1/usr/lpp/java/J5.0/bin:/Z18RD1/usr/lpp/java/J5.0/bin/classic/lib
```

BPXPRM parameters

The SYS1.PARMLIB member BPXPRMxx contains parameters that are used when UNIX System Services is initialized. The parameters that are currently in effect can be displayed by the command /D OMVS,OPTIONS (Example 2-22).

Example 2-22 SYS1.PARMLIB(BPXPRMxx) displayed by /D OMVS,OPTIONS

```
/D OMVS,OPTIONS
RESPONSE=SC66
BPX0043I 09.54.12 DISPLAY OMVS 779
OMVS      0010 ACTIVE          OMVS=(8A)
CURRENT UNIX CONFIGURATION SETTINGS:
MAXPROCSYS      =      4096    MAXPROCUSER      =      32767
MAXFILEPROC     =      65535    MAXFILESIZE     =    NOLIMIT
MAXCPUPTIME     = 2147483647    MAXUIDS       =      1000
MAXPTYS         =      800
MAXMMAPAREA     =      40960    MAXASSIZE     = 2147483647
MAXTHREADS      =    100000    MAXTHREADTASKS =      32767
```

```

MAXCORESIZE      = 2147483647      MAXSHAREPAGES    =      331072
IPCMSGQBYTES     =      262144      IPCMSGQMNUM       =      10000
IPCMSGNIDS       =      4096        IPCSEMNIIDS       =      4096
IPCSEMNIOPS      =      32767       IPCSEMNISEMS      =      32767
IPCshmmpages     =      524287      IPCshmniIDS       =      4096
IPCshmNSEGS      =      1000        IPCshmSPAGES      =      786432
SUPERUSER        = BPXROOT          FORKCOPY          = COW
STEPLIBLIST      =
USERIDALIASTABLE= /etc/ualiastable
PRIORITYPG VALUES: NONE
PRIORITYGOAL VALUES: NONE
MAXQUEUEDSIGS    =      100000      SHRLIBRGNSIZE    = 199999998
SHRLIBMAXPAGES   =      3145728     VERSION          = Z18RD1
SYSCALL COUNTS   = NO               TTYGROUP         = TTY
SYSPLEX          = YES              BRML SERVER      = N/A
LIMMSG           = SYSTEM           AUTOCVT          = OFF
RESOLVER PROC    = DEFAULT
AUTHPGMLIST      = NONE
AUTHPGMLIST      = NONE
SWA              = BELOW
SERV_LINKLIB     =
SERV_LPALIB      =

```

BPXPRMxx also contains the mount commands for your HFS structure. To display which HFS directories are mounted (available), you can use the /D OMVS, F command, as shown in Example 2-23.

Example 2-23 Partial listing of HFSs mounted to UNIX System Services

```

D OMVS,F
BPX0045I 10.01.01 DISPLAY OMVS 833
OMVS      0010 ACTIVE              OMVS=(8A)
TYPENAME  DEVICE -----STATUS----- MODE  MOUNTED  LATCHES
AUTOMNT   36 ACTIVE                RDWR   08/03/2007  L=30
  NAME=*AMD/u                      08.34.33   Q=0
  PATH=/u
  OWNER=SC55      AUTOMOVE=Y CLIENT=N

ZFS        6228 ACTIVE                RDWR   09/27/2007  L=1106
  NAME=CTGUSER.SCSTG714.TMP.HFS    13.12.25   Q=0
  PATH=/ctg/scsctg71/scstg714/tmp
  AGGREGATE NAME=CTGUSER.SCSTG714.TMP.HFS
  OWNER=SC66      AUTOMOVE=Y CLIENT=N

ZFS        6227 ACTIVE                RDWR   09/27/2007  L=1105
  NAME=CTGUSER.SCSTG714.HFS        13.12.24   Q=0

```

```

PATH=/ctg/scsctg71/scstg714
AGGREGATE NAME=CTGUSER.SCSTG714.HFS
OWNER=SC66      AUTOMOVE=Y CLIENT=N
ZFS      6226 ACTIVE                      RDWR  09/27/2007  L=1104
NAME=CTGUSER.SCSTG712.TMP.HFS              13.12.23  Q=0
PATH=/ctg/scsctg71/scstg712/tmp
AGGREGATE NAME=CTGUSER.SCSTG712.TMP.HFS
OWNER=SC66      AUTOMOVE=Y CLIENT=N
ZFS      6225 ACTIVE                      RDWR  09/27/2007  L=1103
NAME=CTGUSER.SCSTG712.HFS                  13.12.22  Q=0
PATH=/ctg/scsctg71/scstg712
AGGREGATE NAME=CTGUSER.SCSTG712.HFS
OWNER=SC66      AUTOMOVE=Y CLIENT=N

```

For more information about these settings, refer to *z/OS V1R8.0 UNIX System Services Planning*, GA22-7800.

CTGDBG

Including a dummy DD statement in the Gateway daemon started task JCL results in extra messages being sent to STDOUT and STDERR (Example 2-24).

Example 2-24 Started task JCL specifying //CTGDBG DD DUMMY

```

//SCSCT711 PROC
//* CTG V7 start proc
// SET CTGHOME='/usr/lpp/cicstg/ctg710'
// SET CTGHLQ='CTG.V7R1M0'
// SET CTGUSR='CICSSYSF.CTG71.STDENV(SCSCT711)'
// SET LEOPTS='/'
//CTG      EXEC PGM=CTGBATCH,REGION=350M,
// PARM='&LEOPTS.&CTGHOME./bin/ctgstart'
//STEPLIB DD DSN=&CTGHLQ..SCTGLOAD,DISP=SHR
//          DD DSN=CICSTS32.CICS.SDFHEXCI,DISP=SHR
//CTGDBG DD DUMMY
//STDOUT  DD SYSOUT=*
//STDERR  DD SYSOUT=*
//STDENV  DD DSN=&CTGUSR.,DISP=SHR
//
//*
```

We found it useful to have the details of the environment, especially the environment variables, which are routed to STDOUT at startup (Example 2-25 on page 75). For ease of maintenance, we placed our environment variables in a STDENV member (CICSSYSF.CTG71.STDENV(SCSCT711)).

Example 2-25 STDOUT with //CTGDBG DD DUMMY specified

```
CTG0826I CTGBATCH Parsed STDENV entry _BPXK_SETIBMOPT_TRANSPORT=TCPIP
CTG0826I CTGBATCH Parsed STDENV entry _BPX_SHAREAS=YES
CTG0826I CTGBATCH Parsed STDENV entry
_CEE_DMPTARG=/ctg/scsctg71/scstg711/logs
CTG0826I CTGBATCH Parsed STDENV entry AUTH_USERID_PASSWORD=NO
CTG0826I CTGBATCH Parsed STDENV entry
CICSCLI=/ctg/scsctg71/scstg711/ctg.ini
CTG0826I CTGBATCH Parsed STDENV entry
CLASSPATH=/usr/lpp/cicstg/ctg710/classes/ctgsamples.jar
CTG0826I CTGBATCH Parsed STDENV entry
CTG_JNI_TRACE=/ctg/scsctg71/scstg711/logs/jni.trace
CTG0826I CTGBATCH Parsed STDENV entry COLUMNS=132
CTG0826I CTGBATCH Parsed STDENV entry DFHJVSYSTEM_00=SCSCPAA1-CICS 3.2
CTG0826I CTGBATCH Parsed STDENV entry DFHJVSYSTEM_01=SCSCPAA4-CICS 3.2
CTG0826I CTGBATCH Parsed STDENV entry DFHJVPIPE=SCSCT711
CTG0826I CTGBATCH Parsed STDENV entry PATH=/bin:/usr/lpp/java/J5.0/bin
CTG0826I CTGBATCH Parsed STDENV entry TMPDIR=/ctg/scsctg71/scstg711/tmp
CTG0826I CTGBATCH Parsed STDENV entry TZ=EST5EDT
CTG0826I CTGBATCH Parsed STDENV entry
CTGSTART_OPTS=-j-Xshareclasses:name=scsctg71
CTG0826I CTGBATCH Parsed STDENV entry CTG_PIPE_REUSE=ONE
CTG0826I CTGBATCH Parsed STDENV entry CTG_MASTER_RRMNAME=SCSCT710
CTG0826I CTGBATCH Parsed STDENV entry CTG_RRMNAME=SCSCT711.IBM.UA
CTG0813I CTGBATCH RLIMIT_AS reports current=357M, system max=2047M
CTG0815I CTGBATCH CWD=/u/ctguser
CTG0817I CTGBATCH PID=50530791
CTG0818I CTGBATCH LOCALE=C
CTG0819I CTGBATCH POSIX=1 USS Version=5
CTG0811I CTGBATCH Runtime env PATH=/bin:/usr/lpp/java/J5.0/bin
CTG0811I CTGBATCH Runtime env CICSCLI=/ctg/scsctg71/scstg711/ctg.ini
CTG0811I CTGBATCH Runtime env TMPDIR=/ctg/scsctg71/scstg711/tmp
CTG0811I CTGBATCH Runtime env TZ=EST5EDT
CTG0811I CTGBATCH Runtime env _BPX_SHAREAS=YES
CTG0820I CTGBATCH Userid=CTGUSER UID=601 GID=87679.
CTG0821I CTGBATCH Initial dir=/u/ctguser Initial program=/bin/sh.
CTG0825I CTGBATCH Spawned child PID=33753734
```

LEOPTS

The LEOPTS parameter on CTGBATCH allows Language Environment® override options to be passed to Language Environment. If `// SET LEOPTS='/'` in Example 2-24 on page 74 is replaced with the following, then the Language Environment runtime options and storage report is routed to STDERR:

```
// SET LEOPTS='RPTOPTS(ON),RPTSTG(ON)/'
```

Tip: The PARM string is limited to 100 bytes. If you find that you need to code very long property strings, you can use the CTGSTART_OPTS environment variable in STDENV.

The df . command

Issuing the `df .` command on a UNIX System Services command line gives information about the mount table entry for the current HFS.

Example 2-26 Use of the df . command to show file system mount information

```
CICSRS6:/ctg/scsctg71/scstg711/tmp: >df .
Mounted on      Filesystem      Avail/Total      Files      Status
/ctg/scsctg71/scstg711/tmp (CTGUSER.SCSTG711.TMP.HFS) 21276/21600    4294967270
Available
CICSRS6:/ctg/scsctg71/scstg711/tmp:
```

Changing the code page for CICS TG messages

We installed the CICS TG using the IBM-1047 code page. It is possible to convert the installation to another code page by running the following command:

```
<install_path>/bin/ctgmsgs <language code> <code set>
```

The user ID that runs this command needs write permission to the `<install_path>/bin` directory. Converting the installation to another code page will not affect messages used by the CTGBATCH program.

To display the supported language and code set combinations, issue the following command (Example 2-27):

```
<install_path>/bin/ctgmsgs -?
```

Example 2-27 Output of ctgmsgs -?

This utility is used to change the language of user messages.

```
-----
      Language      Locale      Code Set
      -----
en   US English    En_US      IBM-1047
```


ja	Japanese	Ja_JP	IBM-939
		Ja_JP.IBM-930	IBM-930
zh	Simplified Chinese	Zh_CN	IBM-935

The syntax of this command is: ctgmsgs <Language> <Code Set>

Error with bad permissions on the directory

We tried to start the Gateway daemon when we had not set the correct permissions to allow it to traverse the /cicstg/ctg710/config directory. The result was that the Gateway daemon failed to start and messages shown in Example 2-28 were written to the Gateway daemon SDSF job log.

Example 2-28 Starting Gateway daemon with incorrect permissions

```
IEF403I SC SCT711 - STARTED - TIME=11.13.36 - ASID=0082 - SC66
ICH408I USER(CTGUSER ) GROUP(CICS ) NAME(PHIL WAKELIN ) 008
/ctg/scsctg71/scsct716/ctg.ini
CL(FSOBJ ) FID(0000000C0000000C0000000000000000)
INSUFFICIENT AUTHORITY TO OPEN
ACCESS INTENT(R--) ACCESS ALLOWED(OTHER ---)
EFFECTIVE UID(0000000601) EFFECTIVE GID(0000087679)
BPXM023I (CTGUSER) CTG6400I SC SCT711 CICS TRANSACTION GATEWAY IS
STARTING
BPXM023I (CTGUSER) CTG6511I SC SCT711 GATEWAY DAEMON HAS SHUT DOWN
```

We needed to set execute permission on the directories for our group using the ISHELL command, specifying the relevant directory, and by selecting **Directory** → **Attributes** → **Edit** → **Owning Group**, as shown in Figure 2-6 on page 37.

Listing libraries in the LINKLIST

We found it helpful to check the libraries in the LINKLIST using the MVS command /D PROG,LNKLST, which gave the results in the log shown in Example 2-29.

Example 2-29 Displaying libraries in the LINKLIST

```
D PROG,LNKLST
CSV470I 16.36.01 LNKLST DISPLAY 141
LNKLST SET LNKLSTLA LNKAUTH=LNKLST
ENTRY APF VOLUME DSNAME
56 A CICS31 SYS1.CICSTS32.CICS.SDFHLINK
57 A CICS31 CICSTS32.CICS.SDFHEXCI
```

SDSF PS command

The UNIX system services processes running on MVS can be displayed using the SDSF command PS (see Example 2-30). This is particularly useful because it shows the number of the process ID (PID) of each process, the address space to which they belong and the parent PID (PPID), and the command that each process is running. More information about this command can be found in the SDSF help panels.

In Example 2-30, you can see that there are three processes (16977427, 16977425, 67309245) running in our Gateway daemon address space (326). These are the CTGBATCH shell launcher (16977425), a UNIX shell (16977427), and the JVM running the Gateway daemon (process 67309245).

Note: Before issuing a PS command, select an individual Gateway started task using a SDSF PREFIX command.

Example 2-30 SDSF PS command output

PREFIX=SCSCT711 DEST=(ALL) OWNER=* SYSNAME=*									
NP	JOBNAME	JobID	Status		Owner	State	CPU-Time		PID
	SCSCT711	STC11618	WAITING FOR CHILD		CTGUSER	1W	7.14		16977427
	SCSCT711	STC11618	FILE SYS KERNEL WAIT		CTGUSER	1F	7.14		16977475
	SCSCT711	STC11618	RUNNING		CTGUSER	HR	7.14		67309245
PPID ASID ASIDX LatchWaitPID Command									
16977475	326	0146		/bin/sh					
1	326	0146		CTGBATCH					
16977427	326	0146		java -Xmx128M -Xms128M					

2.8.2 Tracing

CICS TG on z/OS provides three types of trace:

- Gateway trace
- JNI trace
- EXCI trace

The Gateway daemon and JNI traces can be started statically (where the trace parameters are specified at startup) or dynamically (where the trace parameters are specified during the normal operation of the Gateway daemon).

The EXCI trace can be formatted from a dump of the Gateway daemon address space, or routed to the MVS Generalized Trace Facility (GTF).

Gateway trace

The Gateway trace captures Gateway daemon requests and responses.

Static trace (Gateway daemon)

You can specify the following parameters on the **ctgstart** command when starting the Gateway daemon:

- ▶ **trace=on** (enables trace)
- ▶ **tfile=<path name>** (specifies the destination of the Gateway trace)

The Gateway daemon must be stopped and restarted in order to change these settings. Setting the trace options statically is useful on occasions when you need to trace Gateway daemon initialization.

Dynamic trace (Gateway daemon)

With CICS TG V7.1, it is also possible to administer Gateway tracing dynamically from the SDSF console using the following commands:

- ▶ **/F <job_name>,APPL=TRACE,TLEVEL=0|1|2|3|4**
 - 0: No trace information is output.
 - 1: Exception tracing. Only exceptions are traced.
 - 2: Trace exceptions, and entry and exit of methods.
 - 3: Trace exceptions, some internals, and entry and exit of methods.
 - 4: Full debug tracing.
- ▶ **/F <job_name>,APPL=TRACE,TFSIZE= filesize**

Specifies the maximum size, in kilobytes, of the Gateway trace output file, for example, 50000.
- ▶ **/F <job_name>,APPL=TRACE,DUMPOFSET**

Specifies the offset from which displays of any data blocks start, for example, 512. If the offset is greater than the total length of data to be displayed, an offset of 0 is used.

You cannot use this together with the **fulldatadump** option.
- ▶ **/F <job_name>,APPL=TRACE,TRUNCATIONSZ=**

Specifies the byte at which to stop the hex dump, for example, 2000. It defines the endpoint, not the number of bytes to display. So if on a dump of size 40 you set the dumpoffset to 11, and the truncationsize to 25, you will see 15 bytes (from 11 to 25).

You cannot use this together with the **fulldatadump** option.

► /F <job_name>,APPL=TRACE,FULLDATADUMP

Sets the dumpoffset to 0 and ignores any value specified in truncationsize.

The current trace settings can be displayed by issuing the following command (Example 2-31):

```
/F <job_name>,APPL=TRACE
```

Example 2-31 Results of the command /F SCST714,APPL=TRACE

```
F SCST714,APPL=TRACE
BPXM023I (CTGUSER) 446
CTG8239I Response received from CICS Transaction Gateway
Gateway Daemon trace settings:
  tlevel=0
  truncationsize=80
  dumpoffset=0
  tfile=/ctg/scsctg71/scstg714/logs/ctg.trc
  tfilesize=0
JNI trace settings:
  jnilevel=0
  jnifile=/ctg/scsctg71/scstg714/logs/jni.trace
```

To demonstrate the Gateway trace, we issued the command /F SCST714,APPL=TRACE,TLEVEL=2 (Example 2-32) and used the supplied sample EciB1 to send a request to the Gateway daemon.

Example 2-32 Activating Gateway trace with /F SCST714,APPL=TRACE,TLEVEL=4

```
F SCST714,APPL=TRACE,TLEVEL=4
BPXM023I (CTGUSER) 988
CTG8239I Response received from CICS Transaction Gateway
Gateway Daemon trace settings successfully changed:
  tlevel=4
```

The Gateway trace is written to the destination specified in the **tfile** configuration parameter, in our case, /ctg/scsctg71/scstg714/logs/ctg.trc.

Example 2-33 on page 81 shows a sample Gateway trace output of an ECI request for CICS APPLID SCST714 and an error response Rc=-3 (ECI_ERR_NO_CICS).

Example 2-33 Edited version of our Gateway trace

```
09:22:16:198 Worker-59: S-C: CTG6698I After JNI CcicsECI(1) : Rc=-3, LUW=0, Null
stripped commarea len=
09:22:16:198 Worker-59: .ServerECIRequest:<- ÝgetVersion" = 7344128
09:22:16:198 Worker-59: .ServerECIRequest: + Not using JNI null stripping. JNI
returned data len 0
09:22:16:198 Worker-59: .ServerECIRequest:<- ÝsetInboundDataLength()" = false
09:22:16:198 Worker-59: .ServerECIRequest: + execute(), CICS RC = -3
09:22:16:198 Worker-59: .ServerECIRequest:<- Ýexecute(), Normal Cics_Rc" = false
09:22:16:198 Worker-59: .ServerECIRequest: + getCTGXid() =

09:22:16:198 Worker-59: .Class: + syncTxnCurrentCount(-)=0
09:22:16:198 Worker-59: .ServerECIRequest:<- ÝgetGatewayRc" = 0
09:22:16:198 Worker-59: .ServerECIRequest:-> ÝsetFlowType" (3)
09:22:16:198 Worker-59: .Class: + incrementRequestCount requestCount=521,700,
incrementRequestCount intervalRequestCount=6
09:22:16:198 Worker-59: .ServerECIRequest:-> ÝisCleanupRequest()" ()
09:22:16:198 Worker-59: .ServerECIRequest:<- ÝisCleanupRequest()" = false
09:22:16:198 Worker-59: .ServerECIRequest:-> ÝgetCleanupRequests()" ()
09:22:16:198 Worker-59: .ServerECIRequest:-> ÝreturnCallType()" ()
09:22:16:198 Worker-59: .ServerECIRequest:<- ÝreturnCallType()" = 1
09:22:16:198 Worker-59: .ServerECIRequest:<- ÝgetCleanupRequests()" = 0
09:22:16:198 Worker-59: .TCPHandler:-> ÝsendReply"
(com.ibm.ctg.server.ServerECIRequest@7b627b62)
09:22:16:198 Worker-59: .ServerECIRequest:<- ÝgetFlowType" = 3
09:22:16:198 Worker-59: .ServerECIRequest:-> ÝwriteObject()" (java.io.DataOutput
Stream@3aea3aea)
09:22:16:198 Worker-59: .ServerECIRequest:<- ÝgetVersion" = 7344128
09:22:16:198 Worker-59: .ServerECIRequest:<- ÝgetCallTypeString()" = ECI_SYNC
09:22:16:198 Worker-59: .ServerECIRequest:<- ÝgetCicsRcString()" = ECI_ERR_NO_CICS
09:22:16:198 Worker-59: S-C: CTG6721I ECIRequest: Call_Type = ECI_SYNC, Cics_Rc =
ECI_ERR_NO_CICS, Abend_Code = , Luw_Token = 0, Message_Qualifier = 0.
```

JNI trace

The JNI trace captures information about ECI requests and the EXCI flows that the Gateway daemon initiates.

Static trace (JNI)

The following environment variables can be specified in STDENV to enable JNI tracing at startup. This is useful if you need to investigate JNI problems that occur during Gateway daemon startup:

```
CTG_JNI_TRACE=<pathname>
CTG_JNI_TRACE_ON=YES
```

Dynamic trace (JNI)

The JNI trace can also be started and stopped dynamically using the SDSF MODIFY command:

```
/F <job_name>,APPL=TRACE,JNILEVEL=0|1
```

Where:

- ▶ 0: No trace information is output.
- ▶ 1: On.

Tips: The Gateway and JNI trace destinations cannot be set dynamically. If you do not define specific destinations for the Gateway and JNI traces the output written to STDERR. If both traces are written to STDERR, trace entries are interleaved, making it easier to diagnose problems.

More than one modify command can be issued at one time, for example, the command /F SCSC714,APPL=TRACE,JNILEVEL=1,TLEVEL=2 sets both JNI and Gateway tracing on.

Example 2-34 shows sample JNI trace output of an ECI request for CICS APPLID SCST714 and an EXCI response 8, EXCI reason 203 (NO_CICS).

Example 2-34 Edited version of our JNI trace

```
CTG6801I Performed parameter conversion.  parameter name = jstrServer, parameter
value = SCSCPAA4.
CTG6801I Performed parameter conversion.  parameter name = jstrProgram, parameter
value = EC01      .
CTG6802I Input commarea information.  commarea length = 18, commarea address =
35f5c6e0.
CTG8602I JNI Method zos_SetContext entry
CTG8604I JNI Method zos_SetContext exit (rc = 0).
CTG9251I Entering function 'zos_StatsAddServer'
CTG9297T Variable 'server' = 'SCSCPAA4'
CTG9251I Entering function 'zos_StatsFindServer'
CTG9297T Variable 'servername' = 'SCSCPAA4'
CTG9277I Exiting function 'zos_StatsFindServer' rc = '897049000'
CTG9277I Exiting function 'zos_StatsAddServer' rc = '897049000'
CTG9251I Entering function 'allocateEXCIpipe'
CTG9252I Exiting function 'allocateEXCIpipe' (return code '0')
CTG9251I Entering function 'baseEXCIpipeOperation'
CTG9276I Variable 'call_type' = '3'
CTG9252I Exiting function 'baseEXCIpipeOperation' (return code '0')
CTG6822E EXCI function error.  Function Call = 3, Response = 8, Reason = 203,
Subreason field-1 = 0x5c, subreason field-2 = 0x00, Cics_Rc = -3.
```

```
CTG9251I Entering function 'baseEXCIpipeOperation'  
CTG9276I Variable 'call_type' = '5'  
CTG9252I Exiting function 'baseEXCIpipeOperation' (return code '0')  
CTG6870I EXCI Open pipe gave a Retryable Response. Allocate,open will be retried a  
further 5 times.  
CTG9251I Entering function 'allocateEXCIpipe'  
CTG9252I Exiting function 'allocateEXCIpipe' (return code '0')  
CTG9251I Entering function 'baseEXCIpipeOperation'  
CTG9276I Variable 'call_type' = '3'  
CTG9252I Exiting function 'baseEXCIpipeOperation' (return code '0')
```

We see that the Gateway daemon attempted to call program EC01, but received an error. Because this is a retrievable action (Response code = 8), the request is retried a further five times before the response Rc=-3 is passed back to the application.

Note: Details of the EXCI response and reason codes can be found in the *CICS External Interfaces Guide*. Subreason codes are also documented in hex in the DFHIRSDS member of SDFHMAC.

EXCI trace

The CICS TG writes trace entries to the EXCI trace. The trace entries are in the CICS trace EXCI format, so the trace entries in a dump can be printed using a standard IPCS utility.



Configuring OMEGAMON XE for CICS TG on z/OS

This chapter describes the IBM Tivoli OMEGAMON XE for CICS TG on z/OS V4.1.0 installation and configuration process. It also provides you with planning information that will be useful during the configuration process. Included in this chapter is an outline of the publications for use with the installation and configuration process.

The following topics are discussed:

- ▶ Preparing for the Installation
- ▶ Installing OMEGAMON XE for CICS TG on z/OS
- ▶ Configuring OMEGAMON XE for CICS TG on z/OS
- ▶ Testing the configuration

The information in this chapter should be used as a supplement to the publications delivered with the IBM Tivoli OMEGAMON XE for CICS TG on z/OS product. This information is not intended to be a replacement for any of the product provided publications.

3.1 Preparing for the installation

In this section, we provide an introduction to the IBM Tivoli OMEGAMON XE for CICS TG on z/OS product. We detail the components available for installation and the software levels used in our configuration. A high-level overview of the IBM Tivoli Monitoring infrastructure, of which OMEGAMON XE for CICS TG on z/OS is part of, is included.

3.1.1 Software checklist

In this section, we provide a comprehensive list of all components that may be required for your installation of OMEGAMON XE for CICS TG on z/OS and discuss the levels of software we used as part of our installation.

Software components

The product installation materials you received include z/OS FMIDs, either on tape or in electronic format, physical CDs or electronic CD images that are necessary for some core product functions, and publications on CD as well as in hardcopy or electronic form.

The mandatory components to install are the monitoring agent product files on z/OS and z/OS Configuration Tool to help with post-installation configuration of the runtime environment.

- ▶ z/OS FMIDs provided either on tape or electronic format include:
 - HKGW410 (5698A9300) OMEGAMON XE for CICS TG on z/OS v4.1.0
 - HKCI310 (5608A41CC) Configuration Assistance Tool v3.1.0

Additional Product Package Contents (listed in installation order) are:

- ▶ z/OS FMIDs provided either on tape or electronic format
 - HKDS610(5608A2800) Tivoli Enterprise Monitoring Server on z/OS v6.1.0
 - HKLV610 (5608A41CE) ITMS:Engine v6.1.0
- ▶ Distributed system - CD-ROM or electronic download package contents:
 - LCD7-0901 DB2® UDB Enterprise Server Edition 8.2: Windows CD
 - LCD7-0889 DB2 UDB Enterprise Server Edition 8.2: Linux for S/390® (32-bit) CD
 - LCD7-0890 DB2 UDB Enterprise Server Edition 8.2: Linux for System z (64-bit) CD
 - LCD7-0789 IBM Tivoli Monitoring CD-ROM Vol 1. (Windows) CD

- LCD7-0836 IBM Tivoli Monitoring CD-ROM Vol 5. (Linux on System z - Red Hat and SuSE) CD
- LCD7-0792 IBM Tivoli Monitoring Language Pack CD
- LCD7-0963 IBM Tivoli OMEGAMON Data Files on z/OS CD.
The “data files” CD includes the IBM Tivoli Monitoring application support, which contains the predefined workspaces and situations, online help, expert advice, and OMEGAMON XE data for the Tivoli Enterprise Portal.
- SCD7-0968 IBM Tivoli OMEGAMON XE Documentation CD
- LCD7-0962 IBM Tivoli OMEGAMON XE for CICS TG on z/OS Language Pack CD

Along with the installation materials, you also received a program directory, license booklets, and an Upgrade Road Map for OMEGAMON XE V4.1.0 Monitoring Agents.

Our software levels

We used the following software levels for the z/OS components:

- ▶ z/OS V1R8
- ▶ OMEGAMON XE for CICS TG on z/OS V4.1.0
- ▶ IBM Tivoli Monitoring V6.1.0 Fix Pack 5
- ▶ Configuration Assistance Tool v3.1.0
- ▶ ITMS:Engine V6.1.0
- ▶ IBM SMP/E on z/OS V3.3
- ▶ CICS Transaction Gateway on z/OS V7.1
- ▶ CICS Transaction Server on z/OS V3.2

We used the following software levels for distributed (non-z/OS) components:

- ▶ IBM Tivoli Monitoring V6.1.0 Fix Pack 5
- ▶ IBM Tivoli OMEGAMON Data Files on z/OS V4.1.0
- ▶ DB2 UDB Enterprise Server Edition V8.2
- ▶ IBM SDK on z/OS, Java 2 Technology Edition, V1.4.2

The minimum levels that are supported with OMEGAMOM XE for CICS TG on z/OS are:

- ▶ z/OS V1R7
- ▶ IBM Tivoli Monitoring V6.1.0 Fix Pack 5
- ▶ CICS Transaction Gateway on z/OS V7.0
- ▶ CICS Transaction Server on z/OS V2.2

Important: The following APARs may be required for OMEGAMON XE for CICS TG on z/OS, depending on the version of CICS and CICS TG within your environment and the level of z/OS:

- ▶ OA13628: z/OS V1R7 Currency
- ▶ OA22681: CICS TG V7.1 Currency
- ▶ OA22683: CICS TS V3.2 Currency

Other APARs may have been made available since publication. Check with the OMEGAMON XE for CICS TG on z/OS support Web site for the latest updates:

<http://www-306.ibm.com/software/sysmgmt/products/support/IBMTivoliOMEGAMONXEforCICSTransactionGatewayonZOS.html>

3.1.2 The components of IBM Tivoli Monitoring

IBM Tivoli OMEGAMON XE for CICS TG on z/OS is built upon and forms part of the IBM Tivoli Monitoring (ITM) infrastructure. The infrastructure needed to provide the full monitoring environment is a combination of several vital components that must be installed as part of the OMEGAMON XE for CICS TG on z/OS installation, if not already done so as part of the configuration of another ITM product, such as OMEGAMON XE for CICS on z/OS.

Figure 3-1 on page 89 shows the core IBM Tivoli Monitoring components.

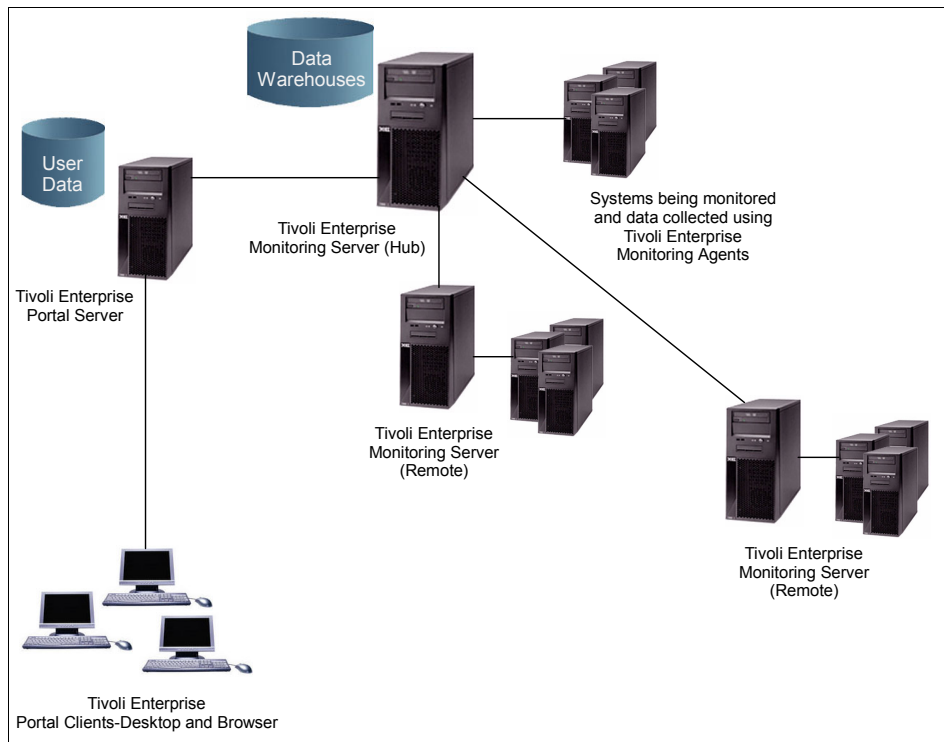


Figure 3-1 IBM Tivoli Monitoring components

► Tivoli Enterprise Monitoring Server (TEMS)

The Tivoli Enterprise Monitoring Server is the key component on which all other architectural components depend directly.

The TEMS acts as a collection and control point for alerts received from Tivoli Enterprise Monitoring Agents (such as OMEGAMON XE for CICS TG on z/OS) and collects their performance and availability data. The TEMS is responsible for tracking the heartbeat request interval for all monitoring agents connected to it. It also stores, initiates, and tracks all situations and policies, and is the central repository for storing all active conditions on every monitoring agent.

The TEMS can be installed on any of the supported operating system platforms (z/OS, UNIX, Linux, and Windows). A single hub TEMS is required in every ITM environment. Other TEMS types that can optionally be created are Remote and Standby. In our environment setup, we installed our hub TEMS on Linux on System z and we installed one remote TEMS on z/OS.

► Tivoli Enterprise Portal Server (TEPS)

The Tivoli Enterprise Portal Server is a repository for all graphical presentation of monitoring data. The TEPS provides the core presentation layer, which allows for retrieval, manipulation, analysis, and reformatting of data. It manages this access through user workspace consoles and administers user access.

The TEPS can be installed on any of the supported operating system platforms (UNIX (AIX only), Linux, and Windows). At least one TEPS is required in every ITM environment. In our environment setup, we installed our TEPS on Windows.

► Tivoli Enterprise Portal (TEP) client

The Tivoli Enterprise Portal client is a Java-based graphical user interface that connects to the Tivoli Enterprise Portal Server to view all monitoring data collections. It is the user interaction component of the presentation layer. The TEP client brings all of these views together in a single window so you can see when any component is not working as expected. The client offers two modes of operation: a Java desktop client and as a downloadable applet in an HTTP browser.

The browser TEP client can only be launched from an Internet Explorer® browser on Windows, or can be installed as a client application on a workstation.

The desktop TEP client can be installed on any of the supported operating system platforms (UNIX (AIX only), Linux, and Windows). In our environment setup, we installed our desktop TEP clients on Windows.

► Tivoli Enterprise Monitoring Agents (TEMA)

The Tivoli Enterprise Monitoring Agents (also referred to as monitoring agents) are installed on the system or subsystem requiring data collection and monitoring.

The agents are responsible for data gathering and distribution of attributes to the TEMS, including initiating the heartbeat status. These agents test attribute values against a threshold and report these results to the monitoring servers. The Tivoli Enterprise Portal displays an alert icon when a threshold is exceeded or a value is matched. The tests are called situations.

Agents exist for the purpose of monitoring operating systems and individual applications. OMEGAMON XE for CICS TG on z/OS is an example of an agent developed for monitoring CICS TG on z/OS.

In our environment setup, we installed the following OMEGAMON XE agents:

- OMEGAMON XE for CICS TG on z/OS V4.1.0

In our environment, this agent was configured to run in its own address space and it reported to the remote TEMS on z/OS.

- OMEGAMON XE for CICS on z/OS V4.1.0

In our environment, this agent was configured to run in its own address space, and it reported to the remote TEMS on z/OS.

- OMEGAMON XE on z/OS V4.1.0

In our environment, this agent was configured to run in the remote TEMS on z/OS address space.

► Tivoli Data Warehouse (TDW)

The Tivoli Data Warehouse is a database created for the storage of long-term historical data collected by the monitoring agents.

In addition to the TDW database itself, the warehousing components consist of one or more Warehouse Proxy agents (WPA) and an optional Summarization and Pruning agent (S&P). The WPA is responsible for the task of receiving and consolidating all historical data collections from the individual agents to store in the TDW database. This occurs at every warehouse interval that is defined by the user, typically either once an hour or once a day.

The S&P agent is responsible for summarizing the raw data located in the TDW database based on the configuration settings specified in the Historical Collection Configuration panel of the TEP client. It creates the corresponding summarized tables in the TDW database. It is also responsible for pruning the data in the TDW database based on the configuration settings in the Historical Collection Configuration panel of the TEP client. The S&P agent is scheduled to execute once a day at a time chosen by the user.

Currently, the TDW is only supported on a Windows platform. In our environment setup, we installed DB2 UDB V8.2 to act as the database. We installed a single WPA and S&P agent on the same Windows machine to collect history from both the monitoring agents through the hub and remote TEMS.

3.2 Installing OMEGAMON XE for CICS TG on z/OS

In this chapter, we identify the steps required to install OMEGAMON XE for CICS TG on z/OS, including important topological considerations and a brief overview of the installation process. We also suggest publications that may be of use in assisting the product installation.

Installation of OMEGAMON XE for CICS TG on z/OS consists of two parts:

- ▶ Installation of the z/OS SMP/E FMIDs
- ▶ Installation of the distributed components using the CD-ROMs supplied in the product package

3.2.1 Installation considerations

Prior to starting your installation, you must first decide where you want to install your hub Tivoli Enterprise Monitoring Server. If you chose to install it on a z/OS system, then you will want to first install and then configure your z/OS SMP/E FMIDs. Alternatively, if you chose to install the hub TEMS on a distributed platform, you should start your install with distributed components using the CD-ROMs from the package.

If you have not already done so, you should review the recommended maintenance found at the following URL:

<http://www-1.ibm.com/support/docview.wss?rs=2366&uid=swg21260251>

The OMEGAMON XE for CICS TG on z/OS support site URL is:

<http://www-306.ibm.com/software/sysmgmt/products/support/IBMTivoliOMEGAMONXEforCICSTransactionGatewayonZOS.html>

From here you can access links to the Tivoli Information Center and the current Preventive Service Planning (PSP) information. As noted in the product program directory, the PSP Upgrade name is OMXEGW410.

3.2.2 Installing product files

For the purpose of this book, we will assume that the installation has already been completed, that is, the SMP/E package installation on z/OS, as well as the applicable distributed components from CD or electronic download, have been installed. For assistance with installation, consult the program directory listed in 3.2.3, “Suggested publications” on page 94.

All installation of the applicable application support for the monitoring agent on distributed platforms is contained in the data files CD. On Windows, this is an InstallShield installation. You should install application support for all ITM products that will be used in the monitoring of CICS TG. Figure 3-2 shows the application support window from the data files CD. In this installation, we have selected to install application support for the TEMS, TEPS and Desktop TEP client components.

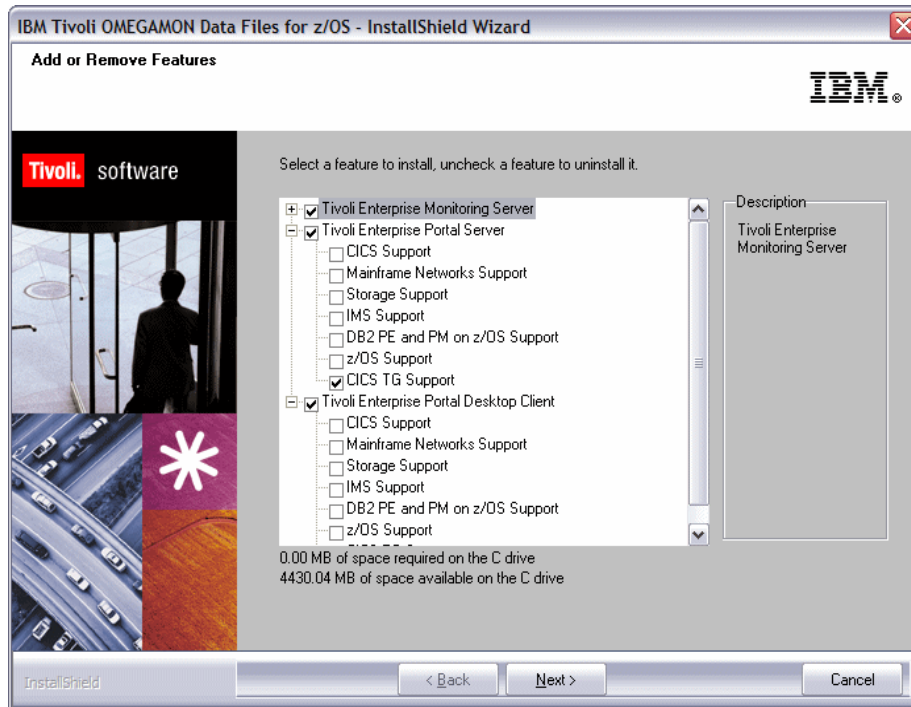


Figure 3-2 Installing OMEGAMON application support on Windows

For further details on our installation, see 4.3, “OMEGAMON XE configuration” on page 127.

Figure 3-3 is a diagram of the environment after the CD-ROM and SMP/E package installation had been completed but before beginning the OMEGAMON XE product configuration on z/OS.

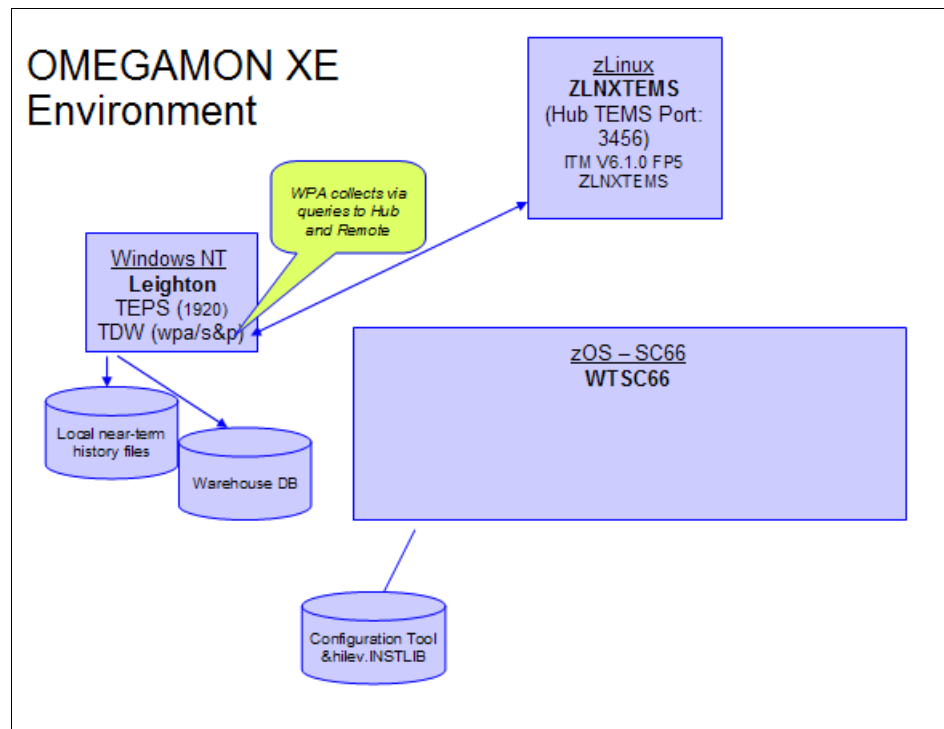


Figure 3-3 OMEGAMON XE environment

After the distributed installation of the CD-ROM was complete, we had a hub TEMS, TEPS, TEP desktop client, and TDW. After the z/OS installation of the SMP/E sysmods was complete, we created our Configuration Tool and were ready to start our agent and remote TEMS configuration on z/OS.

3.2.3 Suggested publications

Here are some suggested publications for use during product installation and configuration planning:

- ▶ *IBM Tivoli Monitoring Services on z/OS Program Directory V6.1.0*, GI11-4105
- ▶ *IBM Tivoli OMEGAMON XE for CICS TG on z/OS Program Directory V4.1.0*, GI11-8079
- ▶ *IBM Tivoli Monitoring Installation and Setup Guide*, GC32-9407

- ▶ *IBM Tivoli OMEGAMON XE for CICS TG on z/OS: Planning and Configuration Guide*, SC23-5962
- ▶ *IBM Tivoli Monitoring: Upgrade Road Map for OMEGAMON XE Version 4.1 Monitoring Agents*, GC32-1980

3.3 Configuring OMEGAMON XE for CICS TG on z/OS

In this chapter, we identify which information should be gathered prior to starting the product configuration and deployment, the steps involved in configuring a runtime environment using the z/OS Configuration Tool, and the configuration required within other ITM components in order to utilize the new monitoring agent. We also identify the publications that can assist with the product configuration.

3.3.1 Configuration considerations

Prior to starting the configuration of an OMEGAMON XE for CICS TG on z/OS agent, you should identify the following key pieces of information:

- ▶ **Location of the CICS TG Dynamic Link Library (DLL):** This library provides the CICS TG statistics API, which is dynamically called during runtime. If you are running multiple versions of CICS TG, use the dataset with the highest CICS TG version to be monitored by a single OMEGAMON XE for CICS TG on z/OS agent.
- ▶ **Gateway daemons names:** You will need to know the names of the Gateway daemons you intend to monitor and the z/OS system name in which they are running.
- ▶ **Statsapi port numbers:** For each of the identified Gateway daemons, find the Gateway daemon *statsapi* port number. The statsapi port number is configured in the *ctg.ini* file using the *statsport* parameter and displayed during the Gateway daemon startup in the STDOUT.

In addition, you should consider the following issues:

- ▶ If you have other OMEGAMON XE products already installed, you will need to decide whether you will be installing OMEGAMON XE for CICS TG on z/OS into an existing runtime environment (RTE) or creating a new RTE.
- ▶ Decide if you will be running the OMEGAMON XE for CICS TG on z/OS agent in its own address space or inside of the TEMS address space. The recommended configuration is to run the agent in its own address space, as this gives the z/OS systems administrator more control over the starting, stopping, and configuration of the monitoring agent.

- ▶ Historical collection requires the allocation of a persistent datastore, which is a set of at least three datasets to hold the sampled data. The size of these files will vary according to the amount of data you want to sample and the frequency of sampling, but they should be large enough to hold the sampled historical data prior to its scheduled transfer to the TDW.
- ▶ If you install the persistent datastore files in the same address space as the OMEGAMON XE for CICS TG on z/OS agent, then when you are configuring historical collection from the TEP client, you should select collection from the agent. This selection is applicable whether the agent is installed to run in its own address space or in the TEMS address space. The only time you would select the TEMS for OMEGAMON XE for CICS TG on z/OS historical collection configuration would be if you had installed the persistent datastore files (KGWHISx) as part of the TEMS configuration and installed the OMEGAMON XE for CICS TG on z/OS agent to run in its own address space.
- ▶ Make a note of the TCP/IP host name and IP address where other IBM Tivoli monitoring components were installed: hub TEMS, remote TEMS, TEPS, and the TDW, including the WPA and S&P agents.
- ▶ If you have an environment where communication between the various monitoring components crosses firewalls, consider the following:
 - The host where the remote TEMS is running will start an outbound connection the machine where the hub TEMS is running and also to the host where the TDW WPA and S&P agents are running.
 - If the OMEGAMON XE for CICS TG on z/OS agent is installed to run in its own address space (which is the default and recommended configuration), the host where the agent is running will need authority to start an outbound connection to the host where the TDW WPA and S&P agents are running.

The following points apply to your CICS TG configuration and should be considered in order to allow monitoring of Gateway daemons to occur:

- ▶ To ensure availability of monitoring information, you should ensure that the OMEGAMON XE agent runs with a z/OS dispatching priority higher than any Gateway daemon being monitored by that agent. Dispatching priority is controlled by the workload manager. In compatibility mode, existing SRM algorithms control the performance of the system. In goal mode, the Tivoli OMEGAMON XE address spaces where the agent is running should have goals set up such that the OMEGAMON XE agent will get the resources it needs to monitor CICS TG daemons running on the same system.
- ▶ The CICS TG product recommends that `_BPX_SHAREAS=YES` be specified in the environment variable file when the Gateway daemon is started through the JCL. Please refer to 2.3.4, “Creating an STDENV file” on page 40 and the description of the environment variables in that chapter for more details. We

also recommend referring to see 2.2.2, “Basic security considerations” on page 30. Setting the value of this variable to YES is required for the OMEGAMON XE agent to locate an active Gateway daemon.

- If you are running WebSphere Application Server on z/OS, you should note that OMEGAMOM XE for CICS TG on z/OS only monitors the Gateway daemon, and so cannot be used when CICS TG is used in local mode. If the monitoring capabilities of OMEGAMON are important to you, then you may want to consider using the CICS TG in remote mode to exploit the monitoring of EXCI and IPIC connections to CICS. See 1.2, “Local and remote modes of operation” on page 12 for information about the different modes of operation when using the CICS TG.

3.3.2 Configuration on z/OS systems

After you have completed your SMP/E installation, you will use the Tivoli z/OS Configuration Tool to create the runtime environments that will contain your OMEGAMON XE for CICS TG on z/OS agent. The *IBM Tivoli OMEGAMON XE for CICS TG on z/OS Planning and Configuration Guide V4.1.0*, SC23-5962 contains detailed information describing this process. In this section, we will show you the two types of configuration methods available from the Tivoli Configuration Tool used on z/OS: the interactive and the batch configuration methods.

Configuration Tool - 3270 interactive method

For the purposes of this book, we will only delve into detail about the configuration options that are unique to the OMEGAMON XE for CICS TG on z/OS product. The sample configuration screens are a subset of the configuration screens presented, but are the only screens that contain configuration options specific to this CICS TG agent.

Starting your agent configuration

Figure 3-4 shows the common agent configuration screen. You must select each of the displayed menu items in the order indicated. The Configuration Tool provides detailed Help information for each of the displays; you can access this help by selecting the F1 key.

```
----- CONFIGURE IBM TIVOLI OMEGAMON XE FOR CICS TG on Z/OS / RTE: SC66 -----
OPTION ==> 2

Perform the appropriate configuration steps in order:

  I Configuration information (What's New)
  1 Register with local TEMS (required if the Agent
    will connect to the TEMS in this RTE.)
  2 Specify configuration parameters

Agent address space configuration:
  3 Specify Agent address space parameters
  4 Create runtime members
  5 Configure persistent datastore (in Agent)
  6 Complete the configuration

Note: This Agent is running in its own Agent address space.

F1=Help  F3=Back  F5=Advanced
```

Figure 3-4 Common agent configuration screen

1. Option 1, *Register with local TEMS*, adds application support files to the TEMS. Here is a summary of when you are required to select this option:
 - If the agent you are configuring will report to a TEMS configured in the same (that is, local) RTE.
 - If your hub TEMS is running on z/OS but the agent reports to a remote TEMS, then you will register the monitoring agent with both the remote TEMS it directly reports to and the hub TEMS.
 - If the agent reports to the hub TEMS directly, then you will only need to register it once. If the hub TEMS is configured in a separate RTE from the agent, then will need to select that RTE for configuration of the OMEGAMON XE agent but you will only select option 1, Register with local TEMS; none of the other configuration items would need to be completed unless you were also going to define an OMEGAMON XE agent in the same RTE as the Hub TEMS.
 - If the agent reports directly to the Hub TEMS but this TEMS is configured in a different RTE from the agent, then you will need to select that RTE for configuration and then only select option 1 (Register with local TEMS) for this configuration. None of the other selected agent configuration items that follow should be performed.

In our configuration, where we wish to report directly to a remote TEMS in the same RTE as the monitoring agent and our hub TEMS is installed on another operating system, we need to register our monitoring agent just once with the remote TEMS.

- In option 2, *Specify configuration parameters*, we specify the configuration options that are unique to OMEGAMON XE for CICS TG on z/OS.

Initially, you are presented with the Gateway daemon Statistics Collection configuration screen, as seen in Figure 3-5. You should enter the information relevant to the Gateway daemons that will be monitored in the specified RTE.

```

----- GATEWAY DAEMON STATISTICS COLLECTION / RTE: SC66 -- Ro
OPTION ==>

CICS TG DLL DSN      ==> CTG.V7R1M0.SCTGDLL

Specify CICS TG DAEMONS to collect statistics from:

Actions: A Add (after), U Update, D Delete

Action  Gateway daemon  Statistics  CTG Trace
-----  Job name      Port number  Level
-----  -----
-         SCST711        2981        0
-         SCST712        2982        0
-         SCST714        2984        0
F1=Help  F3=Back  F5=Adv  F7=Up  F8=Down

```

Figure 3-5 OMEGAMON XE product specific configuration menu

Once you have completed entering the requested information, you can enter PF3 to return to the prior configuration display or you can enter PF5 to view the advanced configuration options menu, as shown in Figure 3-6.

```

----- SPECIFY ADVANCED CONFIGURATION PARAMETERS -----
OPTION ==>

SAPI client sessions      ==> 32      (1-99999)
SAPI client interval      ==> 120     (1-99999 seconds)
SAPI client loop detect   ==> 10      (1-99999 seconds)
SAPI client session timeout ==> 3      (1-99999 seconds)
SAPI client message       ==> LOG     (LOG, WTO, LOGWTO)
SAPI client message output ==> X      (Jes output class)

Agent XMIT number         ==> 00      (00-15)

Enter=Next  F1=Help  F3=Back

```

Figure 3-6 Advanced configuration options menu

The advanced configuration parameters are not required to be changed, but they will be useful to understand. In our configuration, we kept the default values.

Options 3, 4, and 5 are common to all of the OMEGAMON XE monitoring products. These steps are required to be completed if you are configuring the agent to run in its own address space.

3. Option 3, *Specify Agent address space parameters*, is where you will specify the communication values for reporting to the hub, as well the “self-describing” values for configuring up your agent address space, such as security, locale, and so on.
4. In option 4, *Create runtime members*, a batch job is generated that will update the user parameter libraries with the values you have specified as part of configuration options 2 and 3. After submitting this job check, the return code has produced no errors.
5. Option 5, *Configure persistent data store (in Agent)*, will be used to allocate the persistent data store files and start up parameters as part of the agent address space. During the persistent data store configuration, you will be asked to enter JCL jobcard information; it should be noted that this jobcard information is shared by all persistent data store runtime JCL generated for this RTE.

If you have a TEMS defined in this RTE and would like to run the agent in the local TEMS address space, then you will need to select F5 and follow the advanced configuration options. This is not the recommended agent configuration, so it will not be discussed in this book.

6. Option 6, *Complete the configuration*, will guide you through some manual configuration steps that must be performed outside of the Configuration Tool, such as copying the agent started task to your PROCLIB and ensuring libraries are APF authorized. These steps are required to be completed prior to testing your configuration.

Figure 3-7 on page 101 is a diagram of the environment after the our product configuration on z/OS had been completed.

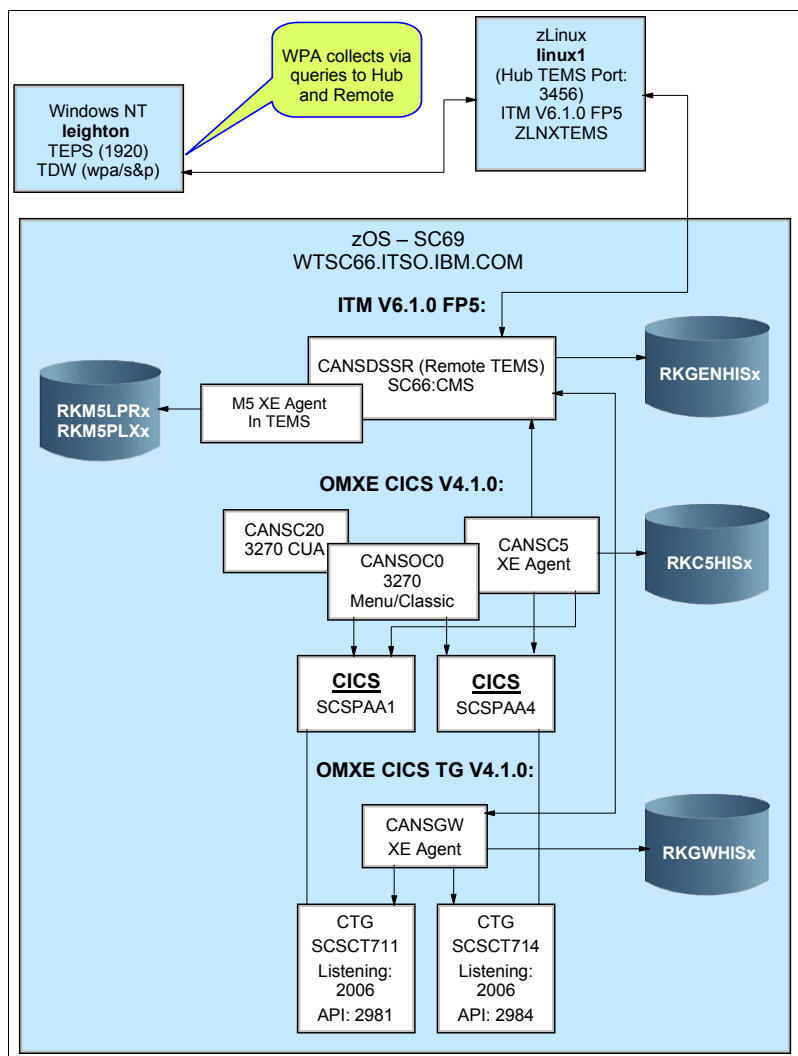


Figure 3-7 Configured OMEGAMON XE environment

Configuration Tool - batch method

The Configuration Tool also offers a batch configuration option. This can be very useful when you have a high number of RTEs where you would like to configure OMEGAMON XE for CICS TG on z/OS agents.

Note: To use batch to configure multiple Gateway daemons for monitoring through OMEGAMON, you must first install the PTF for APAR OA21971.

Example 3-1 shows what a batch parameter member of an RTE configuration with only the OMEGAMON XE for CICS TG on z/OS agent installed might look like.

Example 3-1 Batch parameter member of RTE configuration

```

RTE$310 BEGIN *----- CONFIGURATION TOOL V310 -----*
RTE_DESC          "WTSC66 for CICS"
RTE_TYP           SHARING          * FULL, SHARING or BASE *

** If RTE_TYP is SHARING:
RTE_SHARE         BASE
**
** RTE global defaults:
RTE_HILEV         OMEGAXE
RTE_VSAM_HILEV    OMEGAXE

RTE_VOL           TOTOM1
RTE_VSAM_VOL      TOTOM1
RTE_UNIT          3390
RTE_PDSE          Y
RTE_LOAD_OPTIMIZE N
RTE_LOG_SYSOUT    "T"
RTE_DEBUG_SYSOUT  "S"
RTE_JCL_SUFF      AGTO
RTE_REMOTE        N
RTE_USERMODS      Y
RTE_LOAD_SHARED_LIBS Y
RTE_STC_PREF      CANS
RTE_CMS           N

** (Opt) z/OS System Variable usage values:
RTE_SYSV          Y
RTE_SYSV_NAME     &SYSNAME
RTE_SYSV_VTM_APPL_PREF &SYSNAME.
RTE_SYSV_VTM_NETID USIBMSC
RTE_SYSV_CMS_NAME  "SC66:CMS"
RTE_VTM_CANDLE_NODE AGTOCTD
RTE_SYSV_VTM_APPL_MODEL Y

** (Opt) TCP/IP communications values:
RTE_TCP_HOST      "WTSC66"
RTE_TCP_ADDR      9.12.4.75
RTE_TCP_STC       "*"
RTE_TCP_PORT      3456

```

```

** System procedure libraries:
RTE_SYS_PROCLIB          SYS1.PROCLIB
RTE_SYS_VTAMLST          SYS1.VTAMLST

RTE_PDS_HILEV            OMEGAXE.AGTONLY
RTE_PDS_PROC_PREF        KPDPROC
RTE_PDS_CNT              3
RTE_PDS_VOL              TOTOM1
RTE_PDS_UNIT             3390
RTE_PDS_BU               N
RTE_PDS_EXP              N
RTE_PDS_EXT              N
RTE_PDS_SMS_STOR_CLAS
RTE_PDS_SMS_MGMT_CLAS

RTE$310 END

KGW$410 BEGIN  *--- IBM TIVOLI OMEGAMON XE FOR CICS TG ON Z/OS V410 *

** Values that describe the address space:
KGW_AGT_CONFIG           STANDALONE
KGW_AGT_STC              CANSGW

** Specify communication protocols preference.
KGW_AGT_COMM_PRO1        SNA
KGW_AGT_COMM_PRO2        IPPIPE
KGW_AGT_COMM_PRO3        IP
KGW_AGT_COMM_PRO4
KGW_AGT_COMM_PRO5
KGW_AGT_COMM_PRO6
KGW_AGT_COMM_PRO7

** Values that describe the TEMS to which the agent will connect:
KGW_CMS_LOCAL_CONNECT    N
KGW_CMS_NAME              "SC66:CMS"
** TEMS VTAM Information:
KGW_CMS_VTM_LU62_LOG      CANCTDCS
KGW_CMS_VTM_LU62_LOGTAB   KDSMTAB1
KGW_CMS_VTM_NETID         USIBMSC
**KGW_CMS_VTM_APPL_LLB    &SYSNAME.RMLB                * default *

** (Opt) TEMS TCP/IP info:
KGW_CMS_TCP_HOST          "WTSC66"

```

```

** Protocol port numbers for Agent connection to TEMS:
KGW_CMS_TCP_PIPE_PORT      3456
KGW_CMS_TCP_UDP_PORT       3456

** (Opt) If the Agent requires address translation support:
** KGW_AGT_PIPE_NAME

** (Opt) Agent VTAM and logon info:
**KGW_AGT_VTM_APPL_PREF      &SYSNAME.GW      * default *
**KGW_AGT_VTM_NODE          &SYSNAME.GWN      * default *

** Applids:
**KGW_AGT_VTM_APPL_OPR      &SYSNAME.GWOR      * default *
**KGW_AGT_VTM_APPL_VPO      &SYSNAME.GWVP      * default *
**KGW_AGT_VTM_APPL_NCS      &SYSNAME.GWNC      * default *
**KGW_AGT_VTM_APPL_AA       &SYSNAME.GWAA      * default *

** (Opt) Agent TCP/IP info:
KGW_AGT_TCP_HOST            "WTSC66"
KGW_AGT_TCP_STC             ""
** If the Agent requires network interface list support:
**KGW_AGT_TCP_KDEBLST       ""

** Advanced agent configuration values:
KGW_AGT_KGL_WTO             N
KGW_AGT_WTO_MSG             N
KGW_AGT_STOR_MIN_EXT        150000
KGW_AGT_STOR_DTL_INT_HR     00
KGW_AGT_STOR_DTL_INT_MIN    60
KGW_AGT_FLUSH_INT_HR       00
KGW_AGT_FLUSH_INT_MIN       30
KGW_AGT_VIPA               N
KGW_AGT_ICU_LANG            1

(Opt) Persistent datastore table space allocation overrides:
KGW_PDS_CMS                 N
KGW_PDS_IRA                 Y

KGW_PD                      BEGIN      * Table begin
KGW_PD_ROW
KGW_PD_GRP                  KGW
KGW_PD_CYL                  1
KGW_PD                      END        * Table end *

** CICS TG Dynamic Link Library:

```

```

KGW_CTG_DLL_DSN          CTG.V7R1M0.SCTGDLL

** (Opt) CICS TG Agent XMIT number:
KGW_AGT_XMIT             00

** (Opt) CICS TG Statistics API Client:
KGW_SAPI_CLIENT_SESSIONS 32
KGW_SAPI_CLIENT_INTERVAL 120
KGW_SAPI_CLIENT_MSG_TYPE LOG
KGW_SAPI_CLIENT_MSG_SOUT X
KGW_SAPI_CLIENT_LOOP_DET 10
KGW_SAPI_CLIENT_SESS_TIMO 3

**
** (Opt) Gateway Daemon Statistics Collection table:
**
KGW_SA                     BEGIN                      * Table begin
KGW_SA_ROW
KGW_SA_CTG_DAEMON_JOBNAME SCST711
KGW_SA_CTG_STATS_PORT     2981
KGW_SA_CTG_TRACE_LEVEL    0
KGW_SA_CTG_DAEMON_JOBNAME SCST712
KGW_SA_CTG_STATS_PORT     2982
KGW_SA_CTG_TRACE_LEVEL    0
KGW_SA_CTG_DAEMON_JOBNAME SCST714
KGW_SA_CTG_STATS_PORT     2984
KGW_SA_CTG_TRACE_LEVEL    0
KGW_SA                     END                        * Table end

KGW$410 END

```

Creating batch mode parameters

Use the Create batch mode parameters processing option to export parameters from an existing runtime environment into a library member. You then copy the member and change the image-specific parameters, as required, to configure the runtime environment for its new environment.

You can generate parameter decks for all OMEGAMON XE products in an existing runtime environment, and then copy the information into a new library member to be used during batch mode processing.

Complete the following steps to generate the runtime environment parameters and copy the information into a new library member:

1. From the Configuration Tool Main Menu, select **Configure products** → **Select product to configure** and select the product you are configuring (IBM Tivoli OMEGAMON XE for CICS TG on z/OS).
2. The Configuration Tool displays the Runtime Environments (RTEs) window. Type Z next to the runtime environment you want to replicate and press Enter.
3. From the RTE Utility Menu, select **Create batch mode parameters** and press Enter.
4. Specify the library that receives the batch parameter member generated by the Configuration Tool.
5. The INSTJOBS library is specified by default, and the member name is the same as that of the current runtime environment.
6. Press Enter.
7. Exit the Configuration Tool.
8. Edit the INSTJOBS library and copy the exported library member to a new member name. This new member name is also used as the name of the new runtime environment.
9. Using ISPF Option 2, edit the new library member to reflect the settings specific to the z/OS image of the new runtime environment.

Submitting a Configuration Tool batch job to create a new RTE

Once you have created and modified all of your batch parameter decks, run the Configuration Tool to create the CICATB batch job used to create and configure RTEs using the input supplied in the batch parameter deck.

The Configuration Tool batch job must be submitted on a z/OS system running the same z/OS version, release, and maintenance level as the z/OS system where this RTE will run. The batch job must also be able to write to the DASD where the runtime libraries will reside. The batch parameter decks, Configuration Tool installation libraries (&shlev.INSTLIB), SMP/E target libraries, and BASE RTE libraries may be copied to other systems as needed to meet these requirements.

Do the following steps:

1. From the main Configuration Tool main menu (Figure 3-8 on page 107), select Option 3 Configure products.

```

----- MAIN MENU -----
OPTION ==> 3

Enter the number to select an option:

 1 Set up work environment
 2 Install products or maintenance (for traditional Candle products only)
 3 Configure products
 I Installation information <=== Revised
 S Services and utilities

Installation and Configuration Assistance Tool Version 310.09
(C) Copyright IBM Corp. 1992-2007
Licensed Material - Program Property of IBM

F1=Help F3=Back

```

Figure 3-8 Configuration Tool main menu

2. From the Configure Products menu, select option S, Services and utilities (Figure 3-9).

```

----- CONFIGURE PRODUCTS -----
OPTION ==> S_

Enter the number to select an option:

 1 Set up configuration environment
 2 Select product to configure
 I Configuration information
 S Services and utilities

F1=Help F3=Back

```

Figure 3-9 Configure products menu

3. From the Configuration Services and Utilities menu, select option 2: Create batch mode job. You will see the CICATB JOB CREATED message in the upper right hand corner. Press F1 to display a message telling you where the CICATB JOB was actually created (Figure 3-10).

```
----- CONFIGURATION SERVICES AND UTILITIES -----
OPTION ==> 2_
Enter the number to select an option:
Services:
1  Unlock runtime high-level qualifiers
2  Create batch mode job
Utilities:
3  DEBUG options
4  Display an ISPF table
5  Execute a CLIST in the TKANCUS library
6  Prepare user libraries

Last selected
Date      Time
07/10/08  04:35

F1=Help  F3=Back
```

Figure 3-10 Configuration services and utilities menu

4. Now exit the Configuration Tool and perform the following steps:
 - a. Edit the CICATB member in &shilev.INSTJOBS. Set the SUBMIT parameter to the SCAN and BATCHMEM parameter of your RTE name. This will scan the parameter input for any configuration parameter errors.
 - b. Submit the CICATB job and exit &shilev.INSTJOBS dataset for the CICATB job to execute.
 - c. Review the CICATB job output. Rerun SCAN until there are no errors.
 - d. Edit the CICATB member in &shilev.INSTJOBS. Set the SUBMIT parameter to YES and BATCHMEM parameter to your RTE name or you can set the SUBMIT parameter to NO, which will allow you to selectively submit the generated configuration jobs. Review the CICATB JCL comments for further information.
 - e. Review the CICATB job output.
 - f. If SUBMIT(YES) was selected, then numerous jobs will be submitted to create the RTE. The JCL for these submitted jobs can be found in &shilev.INSTJOBS. Check for RC = 0 or RC = 4 as denoted in the comments for each of the generated jobs.

3.3.3 Configuration on distributed systems

Configuration of most of the distributed ITM components to enable monitoring on CICS TG is performed automatically during the product install. The one exception is adding application support to a TEMS. Recall that a TEMS can reside on many operating systems, such as Windows, UNIX, Linux, or z/OS. The process of adding application support means the copying of product data files into the TEMS's own internal database allowing it to process queries and requests for clients, directing them to the appropriate monitoring agent. In our environment, the hub TEMS was installed on Linux for System z (details on adding the application support to this hub is discussed in 4.3.2, “OMEGAMON XE components installed on Linux for System z” on page 130). Here we describe the process of adding application support to a HUB TEMS, which resides on a different machine than the one on which TEPS is installed.

Important: When adding application support to a TEMS residing on a different machine than where the data files are installed, always ensure the TEMS is online and any firewalls that may lie between the two machines can be crossed.

1. From the Manage Tivoli Enterprise Monitoring Services window, select **Action** → **Advanced** → **Add TEMS application support...**
2. The dialog box shown in Figure 3-11 is displayed. As we are adding application support to a TEMS on z/OS, click the **On a different computer** radio button and click **OK**.

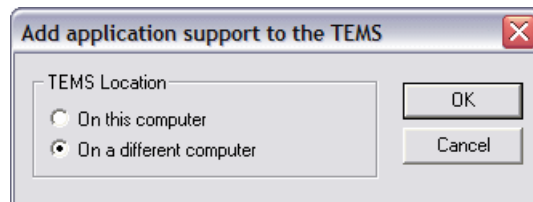


Figure 3-11 Add application support dialog

3. The window shown in Figure 3-12 is displayed. The name of the TEMS we are adding application support to can be found in the KDSENV member of the &rhilev.RKANPARU dataset. We entered the name of our remote TEMS and chose IP.PIPE as the communication protocol with which to send the data. Click **OK**.

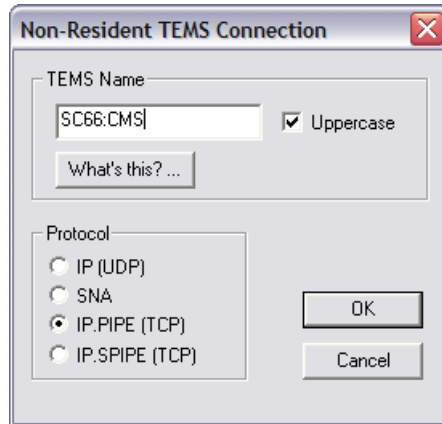
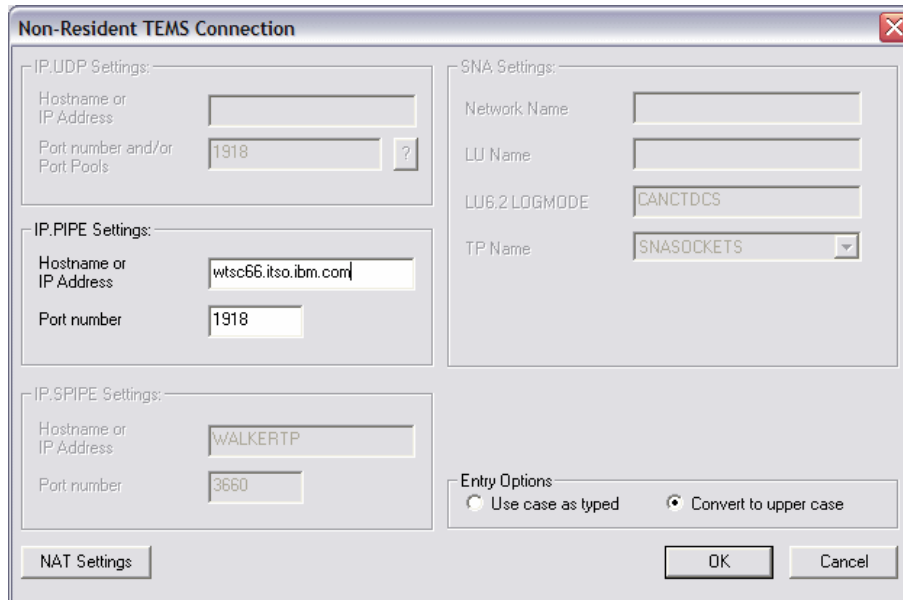


Figure 3-12 Defining non-resident TEMS properties

4. The next window is used to enter the IP address of the machine hosting the TEMS and the port number it is listening on for any requests, as shown in Figure 3-13 on page 111. Enter the details and click **OK**.

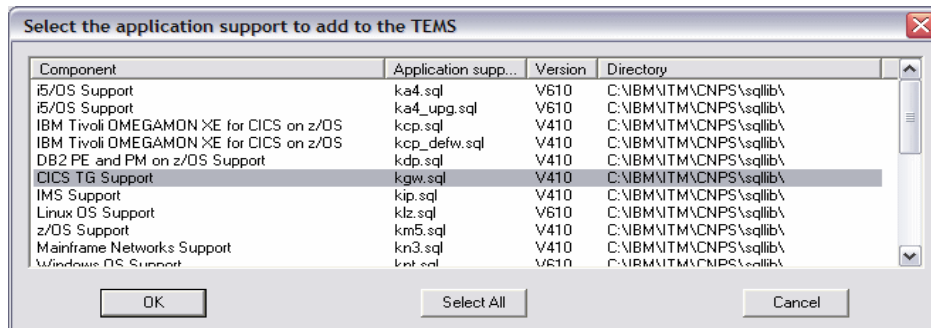


The dialog box is titled "Non-Resident TEMS Connection". It contains several sections for configuration:

- IP.UDP Settings:** Hostname or IP Address (empty), Port number and/or Port Pools (1918).
- IP.PIPE Settings:** Hostname or IP Address (wtsc66.itso.ibm.com), Port number (1918).
- IP.SPIPE Settings:** Hostname or IP Address (WALKERTP), Port number (3660).
- SNA Settings:** Network Name (empty), LU Name (empty), LU6.2 LOGMODE (CANCTDCS), TP Name (SNA SOCKETS).
- Entry Options:** Use case as typed (selected), Convert to upper case (unselected).
- NAT Settings:** A button labeled "NAT Settings".
- Buttons:** OK and Cancel.

Figure 3-13 Defining non-resident TEMS connection properties

- The final dialog box allows you to select which application you want add support for from those installed on this machine. Figure 3-14 shows the selection we had available. Click **CICS TG Support** (and any other agents you want to add support for) and click **OK**.



The dialog box is titled "Select the application support to add to the TEMS". It contains a table with the following data:

Component	Application supp...	Version	Directory
i5/OS Support	ka4.sql	V610	C:\IBM\ITM\CNPS\sqlib\
i5/OS Support	ka4_upg.sql	V610	C:\IBM\ITM\CNPS\sqlib\
IBM Tivoli OMEGAMON XE for CICS on z/OS	kcp.sql	V410	C:\IBM\ITM\CNPS\sqlib\
IBM Tivoli OMEGAMON XE for CICS on z/OS	kcp_defw.sql	V410	C:\IBM\ITM\CNPS\sqlib\
DB2 PE and PM on z/OS Support	kdp.sql	V410	C:\IBM\ITM\CNPS\sqlib\
CICS TG Support	kgw.sql	V410	C:\IBM\ITM\CNPS\sqlib\
IMS Support	kip.sql	V410	C:\IBM\ITM\CNPS\sqlib\
Linux OS Support	klz.sql	V610	C:\IBM\ITM\CNPS\sqlib\
z/OS Support	km5.sql	V410	C:\IBM\ITM\CNPS\sqlib\
Mainframe Networks Support	kn3.sql	V410	C:\IBM\ITM\CNPS\sqlib\
Windows OS Support	knt.sql	V610	C:\IBM\ITM\CNPS\sqlib\

Buttons: OK, Select All, Cancel.

Figure 3-14 Select the application support to add to the non-resident TEMS

- Depending on network traffic and the number of applications you are adding support for, this process may take a few minutes to complete. When finished, a dialog will appear with the results returned from the TEMS. A return code of 0 indicates success.

3.3.4 Historical configuration

In this section, we discuss some considerations for the collection of historical data, that is, a regular sample of the system status so that it can be viewed at some point in the future. Collection of historical data is optional, but it is a very useful task if you want to perform trend analysis or view the system status in the time leading up to a major problem. There are two areas that require configuration if you decide to collect data for historical analysis:

- ▶ Persistent Datastore, where data is stored in the short term after being sampled by the monitoring agent.
- ▶ Tivoli Data Warehouse, where the sampled data is migrated to on a periodic basis, usually every 24 hours, for long-term historical analysis.

The Persistent Datastore

When historical collection is turned on, the data collected is initially stored in the Persistent Datastore (PDS) on z/OS. It will be stored there for the duration of the Warehouse Interval, which is set during TEP Historical Configuration. When configuring the product using the z/OS Configuration Tool, you can adjust the sizes of the PDS files based on your CICS TG workload and collection intervals set in the TEP Historical configuration.

By default, there are three datasets allocated for the PDS (&rhilev.RKGWHIS1, RKGWHIS2, and RKGWHIS3). This minimum of three datasets allows continuous collection of historical collection. The normal case is that one dataset will always be empty, one or more will be full, and one will be *active*. The active dataset is where the latest sample will be written. When the active dataset becomes full, the empty dataset will be activated for continued writing. When the PDS detects that there are no empty datasets left, it will find the one with the oldest data and maintain it. There are two possible strategies:

- ▶ If the BACKUP or EXPORT options were not specified, maintenance is done within the PDS to initialize the dataset so that its status changes from full to empty. This means that any data recorded in this dataset is lost.
- ▶ If the BACKUP or EXPORT options are specified, a job will be run to save the data, and then the dataset will be initialized and marked as empty. The only way that recording would actually stop in the PDS is if the BACKUP or EXPORT was specified but the maintenance jobs fail to do their jobs. In this case, datasets will be taken offline until there are no more available datasets for reading or writing.

We recommend that you allocate enough space to hold 24 hours worth of data; the 24 hours of data must fit into the number of datasets allocated minus 1. This is due to the fact that one must always be kept empty so that when a switch does occur, there will always be an empty dataset to which to switch.

The warehouse code running in the agent to export data will keep track of what it has already warehoused and uses this information to only warehouse new data that was recorded. Therefore, if the TDW is down for any length of time, the data will accumulate in the PDS and be written when the Warehouse Proxy agent is online again. This, of course, is within the limits of how much space is available in the datasets.

If you install the Persistent Datastore files in the same address space as the OMEGAMON XE for CICS TG on z/OS agent, then when you are configuring historical from the TEP client, you should always select the TEMA as the collection location. This configuration is the most typical one and is applicable whether the monitoring agent is installed to run in its own address space or in the TEMS address space. The only time you would select the TEMS for the historical collection location would be if you had installed the Persistent Datastore files (KGWHISx) as part of the TEMS configuration and installed the OMEGAMON XE for CICS TG on z/OS agent to run in its own address space. Figure 3-15 shows our historical collection configuration dialog window in the TEP client. Note that we have set the collection interval to every five minutes, and that the collection location (that is, the location of our Persistent Datastore) is the TEMA.

History Collection Configuration

Select a product
OMEGAMON XE for CICS TG on z/OS V4.1.0

Select Attribute Groups

Group	Collection	Collection Interval	Collection Location	Warehouse Interval	Summarize Yearly	Prune Yearly	Summarize Quarterly	Prune Quarterly	Summarize Monthly	Prune Monthly	Summarize Weekly
OCSTG_Connection_Manager_Threads	Started (2)	5 minutes	TEMA	1 day							
OCSTG_OCS_TS_Region_Details	Started (2)	5 minutes	TEMA	1 day							
OCSTG_OCS_TS_Regions	Started (2)	5 minutes	TEMA	1 day							
OCSTG_Gateway_Daemon	Started (2)	5 minutes	TEMA	1 day							
OCSTG_Region_Overview	Started (2)	5 minutes	TEMA	1 day							
OCSTG_Worker_Threads	Started (2)	5 minutes	TEMA	1 day							

Configuration Controls

Collection Interval: 15 minutes

Collection Location: TEMA

Warehouse Interval: 1 day

Summarization

☐ Yearly
☐ Quarterly
☐ Monthly
☐ Weekly
☐ Daily
☐ Hourly

Pruning

☐ Yearly keep
☐ Quarterly keep
☐ Monthly keep
☐ Weekly keep
☐ Daily keep
☐ Hourly keep
☐ Detailed data keep

Configure Groups Unconfigure Groups Show Default Groups Start Collection Stop Collection Refresh Status

Close Help

Figure 3-15 Historical Collection Configuration dialog in TEP client

The Tivoli Data Warehouse

The Tivoli Data Warehouse uses a DB2, Oracle®, or Microsoft® SQL Server® database to store historical data collected across your environment. You can generate warehouse reports for short-term or long-term data through the Tivoli Enterprise Portal. Warehouse reports provide information about the availability and performance of your monitoring environment over a period of time. You can also use third-party warehouse reporting software to generate reports.

Two specialized agents interact with the Tivoli Data Warehouse:

- ▶ The Warehouse Proxy agent receives data collected by monitoring agents and moves it to the Tivoli Data Warehouse database.
- ▶ The Summarization and Pruning agent provides the ability to customize the length of time for which to save data (pruning) and how often to aggregate granular data (summarization) in the Tivoli Data Warehouse database.

In our configuration shown in Figure 3-15 on page 113, the warehouse interval is set to 1 day, meaning data is exported from the Persistent Datastore to the TDW every 24 hours.

For more details on how we configured our warehousing environment, see 4.3.3, “OMEGAMON XE components installed on Windows” on page 134

3.3.5 Suggested publications

Here are some suggested publications for use during product configuration and customization.

Publications for use during configuration

- ▶ *IBM Tivoli Monitoring Installation and Setup Guide*, GC32-9407
- ▶ *IBM Tivoli OMEGAMON XE for CICS TG on z/OS: Planning and Configuration Guide*, SC23-5962
- ▶ *IBM Tivoli Monitoring: Upgrade Road Map for OMEGAMON XE Version 4.1 Monitoring Agents*, GC32-1980
- ▶ *IBM Tivoli Monitoring: Configuring Tivoli Enterprise Monitoring Server on z/OS*, SC32-9463

Publications for use during customization

- ▶ *IBM Tivoli Monitoring Administrator's Guide*, SC32-9408
- ▶ *IBM Tivoli OMEGAMON XE for CICS TG on z/OS: User's Guide*, SC23-5963
Introduces the features, workspaces, attributes, and predefined situations for the OMEGAMON XE for CICS Transaction Gateway product and supplements the user assistance provided with this product. This document is written for system operators.

You can find these publications by going to the “Tivoli Monitoring and OMEGAMON XE documentation” Information Center, found at:

<http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/index.jsp?topic=/com.ibm.omegamon.cicstg.doc/welcome.htm>

Some suggested publications that will provide additional information to enhance the product configuration and customization experience are as follows:

- ▶ *Tivoli Management Services Warehouse and Reporting*, SG24-7290, found at:
<http://www.redbooks.ibm.com/abstracts/sg247290.html?Open>
- ▶ *Tivoli Data Warehouse Version 1.3: Planning and Implementation*, SG24-6343, found at:
<http://www.redbooks.ibm.com/abstracts/sg246343.html?Open>
- ▶ *Getting Started with IBM Tivoli Monitoring 6.1 on Distributed Environments*, SG24-7143, found at:
<http://publib-b.boulder.ibm.com/abstracts/sg247143.html?Open>
- ▶ *Best Practices for Situation Creation in IBM Tivoli Monitoring V6.1*, found at:
<http://www.redbooks.ibm.com/abstracts/TIPS0617.html?Open>
- ▶ *Implementation Considerations for Pure Versus Sampled Events in IBM Tivoli Monitoring 6.1*, found at:
<http://www.redbooks.ibm.com/abstracts/tips0616.html?Open>
- ▶ *IBM Tivoli Monitoring: Implementation and Performance Optimization for Large Scale Environments*, SG24-7443, found at:
<http://www.redbooks.ibm.com/abstracts/sg247443.html?Open>

3.4 Testing the configuration

This section explains how to verify if your configuration is successful and if your monitoring environment is functioning as expected. We verify that:

- ▶ The remote TEMS can be started.
- ▶ The OMEGAMON XE for CICS TG on z/OS can be started.

We assume that you have started the other components in your monitoring infrastructure, such as the hub TEMS and the Gateway daemons you intend to monitor.

3.4.1 Verify the remote TEMS startup

Start the remote TEMS started task using the following z/OS command:

```
/S CANSDSSR
```

Successful startup will be indicated by the following messages found in the RKLVL0G:

- ▶ This message indicates your agent “self-describing” communications are active:

```
KDSNC004 Bind of local location broker complete=  
ip.pipe:#9.12.4.75.3456.
```
- ▶ This Indicates your agent communications to the Hub TEMS are active:

```
"KDCG_Bind") Using GLB at ip.pipe:#9.12.4.111.3456.
```
- ▶ This indicates the near-term history Persistent Datastore is active:

```
KPDEIMN: Persistent Datastore initialization completed  
successfully.
```
- ▶ Once you start the monitoring agent, as shown in 3.4.2, “Verify the OMEGAMON XE agent startup” on page 117, the following should appear in this log to indicate that the agent is online:

```
"HeartbeatInserter") Remote node <CICSTG:SC66:GWIRA> is ON-LINE.<==  
CICSTG agent is online
```

You will see the following message when situations go true and are no longer true. Note, however, you will only see these messages if you have distributed situations to run at this agent.

```
K041041 Enterprise situation
```


- ▶ You will then see the following logon messages when the Gateway daemons (subnodes) being monitored come online:

```
Logon successful to server SRVR01 user SRVR01
ip.pipe:#9.12.4.111(3456).
```

3.4.2 Verify the OMEGAMON XE agent startup

Start the OMEGAMON XE for CICS TG on z/OS monitoring agent started task using the following z/OS command:

```
/S CANSGW
```

Successful startup will be indicated by the following messages found in the RKLVLLOG:

- ▶ When the near-term history Persistent Datastore is active, the following messages will be issued:
 KPDEIMN: Persistent Datastore initialization completed successfully
- ▶ The following messages indicate that the OMEGAMON XE attach controller has successfully started:
 KGW9000I: ATC INITIALIZATION HAS STARTED
 KGW9950I: ATC HAS LOADED MODULE: KGWATTOA
 KGW9950I: ATC HAS LOADED MODULE: KGWATIOA
 KGW9950I: ATC HAS LOADED MODULE: KGWTDAAO
 KGW9009I: ATC INITIALIZATION HAS ENDED
- ▶ This message indicates that the GWIRA agent thread is registered:
 KGW9002I: IRA REGISTRATION THREAD IS ACTIVE
- ▶ When the GWIRA agent is starting its heartbeat and attempting to connect to the TEMS, you will see the following two messages:
 "ctira_insert_log") KRAIRA000, Starting HEARTBEAT
 "ctira_insert_log") KRAREG000, Connecting to CMS SC66:CMS
- ▶ The following messages indicate the historical situations are starting. Note, however, you will only see these messages if you have configured historical collection for this agent.
 "ctira_insert_log") KRAIRA000, Starting UADVISOR_

- You will see one of each of the following messages in the RKGWSLOG for each of the Gateway daemons that you are monitoring:

```
KGWSPI09I Session Name=SCSCT711 Port=02981 Sysname=SC66  
Jobname=SCSCT711 Token=x"308D94A8" Status=Active  
KGWSPI09I Session Name=SCSCT714 Port=02984 Sysname=SC66  
Jobname=SCSCT714 Token=x"307CA4A8" Status=Active
```



Part 2

Problem determination scenarios

In this part, we provide detailed information about how we used the CICS TG systems monitoring functionality in conjunction with OMEGAMON XE to diagnose a variety of typical customer problems.



Scenario environment

In this chapter, we introduce the environment developed for the scenarios discussed in subsequent chapters. We discuss the topology of the environment, the components used, and how we configured them. We also document the baseline performance levels of the environment ahead of the performance problems discussed in the subsequent scenarios.

4.1 Introducing the environment

The environment developed for the scenarios discussed in the subsequent chapters is shown in Figure 4-1

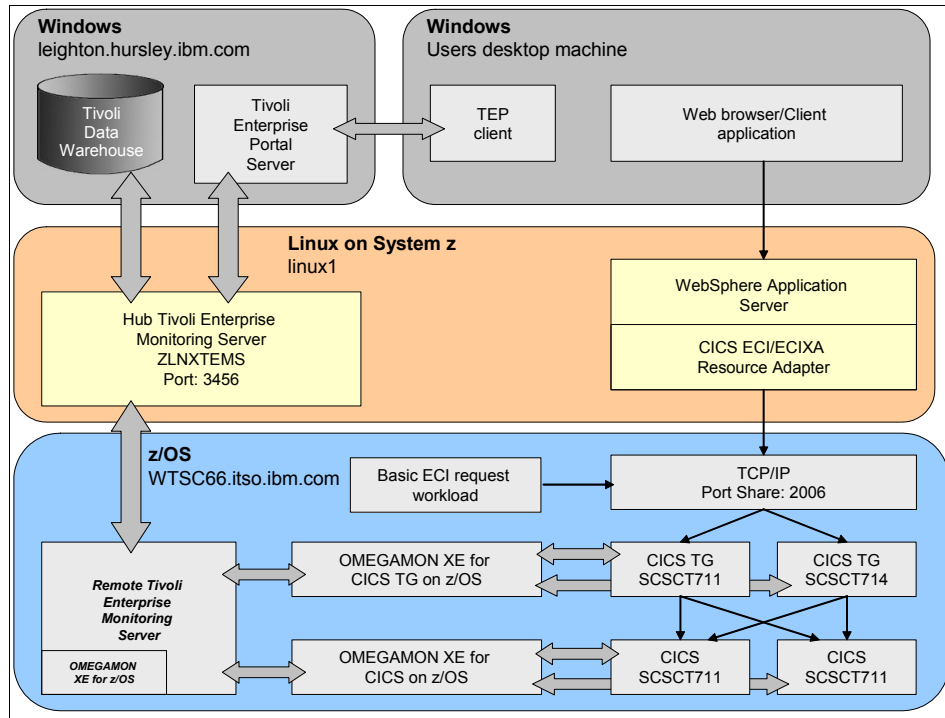


Figure 4-1 Scenario environment overview

Figure 4-1 shows the principle components that make up the environment, including the CICS Transaction Gateway, CICS regions, and OMEGAMON XE agents. The thin lines represent the flow of data requests from client applications through the Gateway daemons to a CICS region. The thick lines represent the flow of data within the monitoring infrastructure, such as a request from a user for monitoring data through the Tivoli Enterprise Monitoring Server to the appropriate OMEGAMON XE agent.

There are two types of workload used in our scenarios. First, we ran a basic set of ECI requests into cloned Gateway daemons using a shared TCP/IP port. Second, we also ran work through an instance of WebSphere Application Server in order to create a transactional workload using XA flows into the CICS Transaction Gateway.

4.1.1 Software checklist

To simulate a typical customer environment, components were installed on various operating systems, such as z/OS, Linux on System z, and Windows. Table 4-1 shows the levels of software used for our environment.

Table 4-1 Software checklist

z/OS	Linux for System z	Windows
z/OS V1.8	SUSE Linux Enterprise Server 10 Service Pack 1	Windows XP Service Pack 2
CICS Transaction Server V3.2	Tivoli Monitoring V6.1.0 Fixpack 5 (Tivoli Enterprise Monitoring Server)	Tivoli Monitoring V6.1.0 Fix Pack 5 (Tivoli Enterprise Portal Server, Tivoli Enterprise Portal, Warehouse Proxy agent, Summarization and Pruning agent)
CICS Transaction Gateway V7.1	Tivoli OMEGAMON Data Files on z/OS V4.1.0	Tivoli OMEGAMON Data Files on z/OS V4.1.0
Tivoli OMEGAMON XE for CICS TG on z/OS V4.1.0	WebSphere Application Server V6.1 Fix Pack 11	DB2 UDB Server V8.1 Fixpack 7
Tivoli OMEGAMON XE for CICS on z/OS V4.1.0		IBM Java SDK on z/OS V1.4.2 SR4
Tivoli OMEGAMON XE on z/OS V4.1.0		
IBM Java SDK on z/OS V5.0		

4.2 CICS TG and CICS configuration

Figure 4-2 shows in detail the configuration of CICS and CICS TG within our environment.

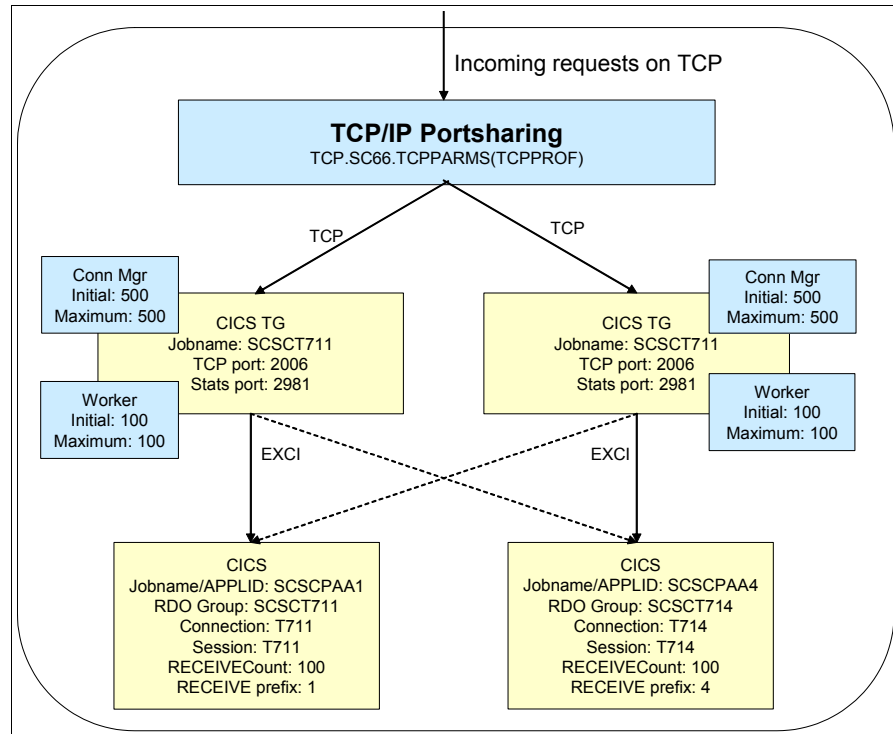


Figure 4-2 CICS and CICS TG configuration

The basic configuration for our environment consisted of identically configured CICS regions connected to two instances of the CICS TG. Each Gateway daemon connects by default to one of the CICS regions (SC SCT711 to SC SCPAA1 and SC SCT714 to SC SCPAA4) but can send work to the other CICS region in the case of failover. This provides a high availability environment capable of enduring a component failure or system slowdown.

4.2.1 CICS TG configuration

We created two Gateway daemons in our environment called SC SCT711 and SC SCT714. Table 4-2 on page 125 details the configuration settings used for these Gateway daemons.

Table 4-2 CICS TG configuration settings

Setting	SCSCT711 value	SCSCT714 value
Host name	wtsc66.itso.ibm.com	
IP address	9.12.4.76	
Stats DLL dataset	CTG.V7R1M0.SCTGDLL	
Job name	SCSCT711	SCSCT714
APPLID	SCSCT711	SCSCT714
NETNAME	SCSCT711	SCSCT714
TCP port	2006	2006
Stats port	2981	2984
Connection Manager threads	<ul style="list-style-type: none"> ▶ Initial: 500 ▶ Maximum: 500 	<ul style="list-style-type: none"> ▶ Initial: 500 ▶ Maximum: 500
Worker threads	<ul style="list-style-type: none"> ▶ Initial: 100 ▶ Maximum: 100 	<ul style="list-style-type: none"> ▶ Initial: 100 ▶ Maximum: 100
XA support	On	On
Health reporting	On	On
Health interval (seconds)	1	1
Stats recording	On	On
Stats interval (HHMMSS)	010000 (1 hour)	010000 (1 hour)
Stats end of day (HHMMSS)	000000 (midnight)	000000 (midnight)

Each Gateway daemon was configured with its own configuration file (ctg.ini). Both specify 2006 for their TCP listener port to allow us to utilize TCP/IP load balancing across a shared port. When TCP/IP port sharing is used, requests for work to be shared between several Gateway daemons are received through a single TCP/IP port. This means that the Java client application will send a request to a single port and TCP/IP will establish a connection with one of several Gateway daemons listening on the same port number. Once a connection is established, the Java application will continue to use the same Gateway daemon connection. For further details on configuring CICS TG and TCP/IP port sharing, see “Implementing TCP/IP port sharing” on page 67.

The stats recording parameter was set to on to enable SMF records to be produced for each stats interval (which we set to hourly), the end of day time (which we set to the default of midnight) plus the lifetime of the Gateway daemon.

4.2.2 CICS configuration

To match our two Gateway daemons, we configured two CICS regions to process the incoming work requests. Table 4-3 details the basic properties of these regions.

Table 4-3 CICS Transaction Server settings

Setting	SCSCPAA1 value	SCSCPAA4 value
Jobname	SCSCPAA1	SCSCPAA4
APPLID	SCSCPAA1	SCSCPAA4
CONNECTION	Name:T711 Protocol: EXCI NETNAME: SCST711	Name: T714 Protocol: EXCI NETNAME: SCST714
SESSIONS	Name: T711 Connection: T711 Protocol: EXCI RECEIVEPFX: 1 RECEIVECOUNT: 200	Name: T714 Connection: T714 Protocol: EXCI RECEIVEPFX: 4 RECEIVECOUNT: 200
RRMS	YES	YES

Each CICS region has a connection definition giving a specific EXCI connection through to a Gateway daemon. As can be seen, CICS region SCSCPAA1 by default communicates with Gateway daemon SCST711 while SCSCPAA4 by default communicates with SCST714. For a specific EXCI connection, the value specified to the Gateway daemon environment variable DFHJVPIPE must be the same as the NETNAME option on the connection definition. We defined the EXCI NETNAME here to match the name of the Gateway daemon.

In order to allow XA transactional support in our environment, we need to configure CICS to register with Resource Recovery Management Services (RRMS). We enabled CICS registration with RRMS by setting the CICS system initialization (SIT) parameter RRMS=YES as an override.

For the purposes of simplicity, no additional security measures were installed or configured. For more details on the configuration of our CICS regions, refer to 2.4, “Configuring CICS TS” on page 52.

4.3 OMEGAMON XE configuration

We used IBM Tivoli Monitoring (ITM) and the OMEGAMON XE products to provide real-time and historical monitoring of the Gateway daemons and CICS regions deployed in this environment. Monitoring of the z/OS LPAR where these applications reside was also provided using OMEGAMON XE on z/OS.

Figure 4-3 shows a detailed view of the ITM configuration used.

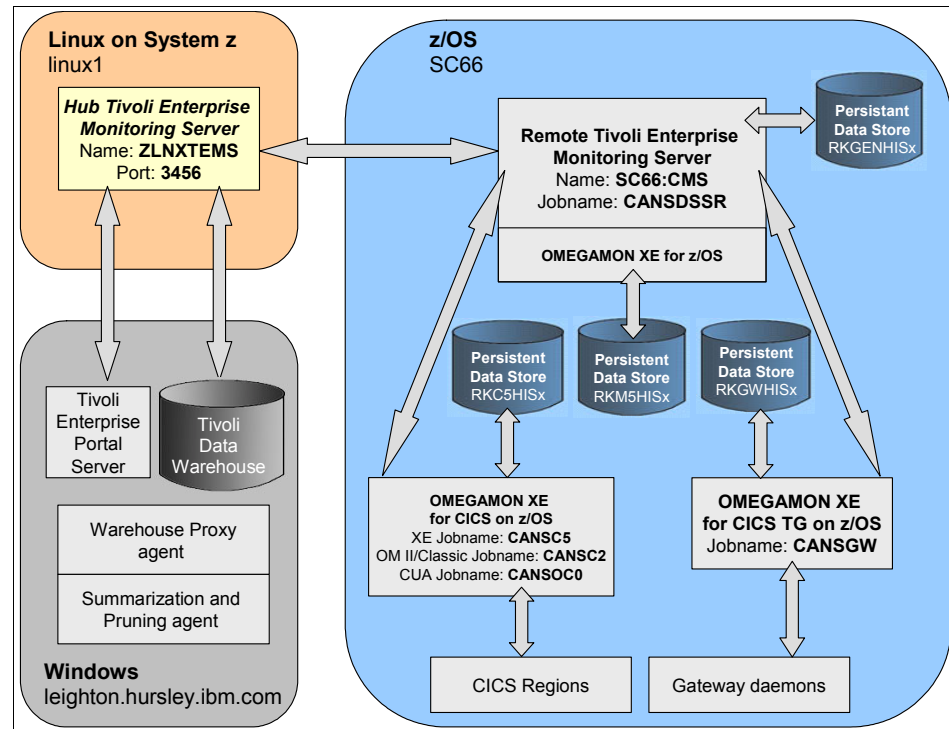


Figure 4-3 Tivoli Monitoring and OMEGAMON configuration

4.3.1 OMEGAMON XE components installed on z/OS

Configuration of the ITM/OMEGAMON XE environment was performed using the z/OS Installation and Configuration Assistance Tool (ICAT).

An individual monitoring agent was created for OMEGAMON XE for CICS TG on z/OS and OMEGAMON XE for CICS on the z/OS LPAR SC66. Both these agents were configured to run in their own address space. Additionally for OMEGAMON XE for CICS, the OMEGAMON II® classic components have been configured, though further discussion of this is beyond the scope of this book.

Note: Although we are not discussing the install and configuration of OMEGAMON XE for CICS, it is important that all components including the OMEGAMON II agent and Common User Access® (CUA®) are configured, otherwise your CICS monitoring agent will not function correctly. OMEGAMON XE for CICS TG on z/OS only has a single XE agent to be configured.

Table 4-4 lists the configuration settings applied to the OMEGAMON XE for CICS TG agent.

Table 4-4 OMEGAMON XE for CICS TG on z/OS agent settings

Setting	Value
Jobname/Started Task	CANSGW
Communication Protocol Priority (for communication with remote TEMS)	1. SNA.PIPE 2. IP.PIPE 3. IP.UDP
Name of Primary TEMS	SC66:CMS
Persistent Data Store group	RKGWHIS
Number in group	3 (default)
PDS storage size	30 cylinders

All other settings for this agent were set to their default values in ICAT. Using the advanced settings, the agent has been configured to communicate with the two Gateway daemons defined above using their stats API port and stats DLL, as detailed in Table 4-2 on page 125.

Table 4-5 lists the configurations settings applied for the OMEGAMON XE for CICS agent.

Table 4-5 OMEGAMON XE for CICS on z/OS agent settings

Setting	Value
Jobname/Started Task	<ul style="list-style-type: none"> ▶ XE agent: CANSC5 ▶ OMII ("Classic") agent: CANSOC0 ▶ Common User Access (CUA): CANSC2
Communication Protocol Priority (for communication with remote TEMS)	1. SNA.PIPE 2. IP.PIPE 3. IP.UDP
Name of Primary TEMS	SC66:CMS

Setting	Value
Persistent Data Store group	RKC5HIS
Number in group	3 (default)
PDS storage size	248 cylinders

Also on this LPAR, a remote TEMS has been configured to control communication between the monitoring agents and the hub TEMS, which will be sending and receiving requests for status data. Within the same address space of the remote TEMS is an OMEGAMON XE on z/OS monitoring agent. This provides monitoring information about the performance of the LPAR itself and we make use of this in the scenarios by employing the Dynamic Workspace Links defined between the monitoring agents.

Note: It is a recommended configuration to have OMEGAMON monitoring agents to connect to a remote TEMS rather than directly to the hub TEMS for reasons of performance and scalability. There can be only one hub TEMS within the monitoring environment but several remote TEMS. Connecting monitoring agents direct to the hub increases the processing the hub must perform in terms of coordinating each agents status. This leads to an overall detrimental effect for the performance of the monitoring environment.

Table 4-6 lists the configuration settings applied for the remote TEMS and OMEGAMON XE on z/OS agent.

Table 4-6 Remote TEMS and OMEGAMON XE on z/OS agent settings

Setting	Value
Jobname/Started Task	CANSDSSR
TEMS name	SC66:CMS
Name of hub TEMS to connect to	ZLNXTMS
Communication Protocol Priority (for communication with hub TEMS)	1. IP.PIPE 2. IP.UDP 3. SNA.PIPE
Persistent Data Store group	► Remote TEMS: RGENHIS ► OMEGAMON XE on z/OS: RKM5LPR (LPAR data) RKM5PLX (PLEX data)
Number in group	3 (default)

Setting	Value
PDS storage size	<ul style="list-style-type: none">▶ Remote TEMS: 36 cylinders▶ OMEGAMON XE on z/OS<ul style="list-style-type: none">686 (LPAR data)195 (PLEX data)

The host name and port numbers needed to make the connection to the hub TEMS are detailed in 4.3.2, “OMEGAMON XE components installed on Linux for System z” on page 130, where we define the hub’s connection properties.

For each of our three OMEGAMON XE monitoring agents and the remote TEMS, we defined a Persistent Data Store (PDS) to be used for holding short-term historical data recorded by the monitoring agent. Each PDS was configured with a storage size estimated to be large enough to hold 24 hours of data, after which point it will be migrated to Tivoli Data Warehouse. Factors to be considered when estimating the storage size are:

- ▶ Number of tables to be collected
The amount of data you can choose to collect can vary from agent to agent. OMEGAMON XE for CICS on z/OS has many more tables than OMEGAMON XE for CICS TG on z/OS that can be collected, so you may wish to allocate more space for data collected from CICS. Also, you do not need to collect data from every table from which a monitoring agent can record data. Knowledge about which tables that you specifically want to store can give a more accurate estimate.
- ▶ Frequency of data collection
The frequency of data collection has a major impact on the amount of data collected. The frequency can be configured to record data from every five minutes up to once an hour.

In our environment, we varied the collection interval and we provided an estimate to allow a large amount of data to be collected prior to warehousing.

4.3.2 OMEGAMON XE components installed on Linux for System z

Each ITM/OMEGAMON environment requires a single hub TEMS to be configured to act as the centralized management point of the system monitoring. The hub TEMS can be deployed on a variety of operating systems, including z/OS, Windows, and several distributions of UNIX. We chose to deploy onto a Linux for System z image, as this is an increasingly popular choice for many users, as it combines the benefit of reducing the total cost of ownership by moving the heavy processing load of the hub TEMS off z/OS while still maintaining the infrastructure benefits of utilizing a System z server.

Installation of the hub TEMS into this environment is a two-step process. First, the base ITM components are installed (that is, the TEMS) and second, we installed the OMEGAMON application support (that is, the data files containing the queries and information related to our OMEGAMON agents) into the TEMS.

ITM/OMEGAMON is installed into Linux for System z using a command-line (non-GUI) based installation script. When installing the base ITM components, we chose to install only the Tivoli Enterprise Monitoring Server (TEMS) and gave it the name *ZLNXTMS*.

When installing the OMEGAMON application support, we chose to install the following:

- ▶ All the application support for the OMEGAMON products found under the menu option for the operating system we were using.
- ▶ All the application support for the OMEGAMON products found under the menu option for Tivoli Enterprise Monitoring Server support.

Note: Although we installed all application support, the products that must be installed for this environment are OMEGAMON XE on z/OS, OMEGAMON XE for CICS on z/OS, and OMEGAMON XE for CICS TG on z/OS. There may also be interim fix packs available containing a roll-up of recent APARs. At the time of configuring our environment, both OMEGAMON XE on z/OS and OMEGAMON XE for CICS on z/OS have interim fix packs available. Check with the appropriate IBM support Web site for the latest updates.

Once the install is complete, the TEMS needs to be configured and the application support applied so that it is able to drive requests to each of the various monitoring agents already configured on z/OS and handle the resulting data being returned. ITM on Linux and UNIX provides a useful script called *itmcmd* that drives the configuration process. It can be found in the */opt/IBM/ITM/bin* folder.

Example 4-1 shows the steps of configuring the hub TEMS using the command */opt/IBM/ITM/bin/itmcmd config -S -t ZLNXTMS*.

Example 4-1 Configuring the hub TEMS

```
linux1:/opt/IBM/ITM/bin # ./itmcmd config -S -t ZLNXTMS
Configuring TEMS...
```

```
Hub or Remote [*LOCAL or *REMOTE] (Default is: *LOCAL): *LOCAL
TEMS Host Name (Default is: linux1): linux1
```

Network Protocol 1 [ip, sna, ip.pipe, or ip.spipe] (Default is: ip.pipe):

Now choose the next protocol from one of these:

- ip
- sna
- ip.spipe
- none

Network Protocol 2 (Default is: none):

IP.PIPE Port Number (Default is: 1918): 3456

Enter name of KDC_PARTITION (Default is: null):

Enter path and name of KDC_PARTITIONFILE (Default is: /opt/IBM/ITM/tables/ZLNXTMS/partition.txt):

Configuration Auditing? [YES or NO] (Default is: YES): YES

Hot Standby TEMS Host Name (Default is: none): none

Enter Optional Primary Network Name or "none" :(Default is: none):

Security: Validate User ? [YES or NO] (Default is: NO): NO

TEC Event Integration Facility? (Default is: NO): NO

Disable Workflow Policy/Tivoli Emitter Agent Event Forwarding? (Default is: NO): NO

... Writing to database file for ms.

Editor for SOAP hubs list

Hubs

##	CMS_Name
1	ip.pipe:Linux1[3456]

A)dd, R)emove ##, M)odify Hub ##, U)serAccess ##, C)ancel, S)ave/exit:
S

... creating config file

"/opt/IBM/ITM/config/linux1_ms_ZLNXTMS.config"

... creating file "/opt/IBM/ITM/tables/ZLNXTMS/glb_site.txt."

... updating "/opt/IBM/ITM/config/kbbenv"

... verifying Hot Standby.

TEMS configuration completed...

linux1:/opt/IBM/ITM/bin #

The hub TEMS can be then started and stopped using the following commands:

- ▶ Starting the hub TEMS:

```
/opt/IBM/ITM/bin/itmcmd server start ZLNXTMS
```

- ▶ Stopping the hub TEMS:

```
/opt/IBM/ITM/bin/itmcmd server stop ZLNXTMS
```

The hub TEMS needs to be started in order to add the application support.

Example 4-2 shows the hub TEMS starting successfully.

Example 4-2 Starting the hub TEMS

```
linux1:/opt/IBM/ITM/bin # ./itmcmd server start ZLNXTMS
Starting TEMS...
TEMS started...
linux1:/opt/IBM/ITM/bin #
```

Example 4-3 shows the steps of adding the application support to the hub TEMS using the command **/opt/IBM/ITM/bin support -t ZLNXTMS gw cp m5**.

Example 4-3 Adding OMEGAMON XE application support to the hub TEMS

```
linux1:/opt/IBM/ITM/bin # ./itmcmd support -t ZLNXTMS gw cp m5
Multi-product support installation: gw
Copying cat and attr data...
Product support installation started...
Info: Seeding with /opt/IBM/ITM/tables/cicatrsq/SQLLIB/kgw.sql
Product support installation completed...
Multi-product support installation: cp
Copying cat and attr data...
Product support installation started...
Info: Seeding with /opt/IBM/ITM/tables/cicatrsq/SQLLIB/kcp.sql
Product support installation completed...
Multi-product support installation: m5
Copying cat and attr data...
Product support installation started...
Info: Seeding with /opt/IBM/ITM/tables/cicatrsq/SQLLIB/km5.sql
Product support installation completed...
Multi-product support installation completed
linux1:/opt/IBM/ITM/bin #
```

Note: Each ITM/OMEGAMON on UNIX and Linux operating systems has a two-letter product code, called AVA codes, and these are used in the adding application support. The codes are:

- ▶ gw OMEGAMON XE for CICS TG on z/OS
- ▶ cp OMEGAMON XE for CICS on z/OS
- ▶ m5 OMEGAMON XE on z/OS

A comprehensive list of ITM/OMEGAMON product codes can be found by looking at the `/opt/IBM/ITM/registry/proddsc.tbl` file.

We then recycled the hub TEMS following the addition of the application support. Once the hub TEMS is online, the remote TEMS we configured on z/OS will make a connection and report the status of the monitoring agents for which it has responsibility.

Tip: ITM on UNIX and Linux operating systems provides a useful command-line utility called CINFO. Using this utility allows you to view which components have been installed on this machine, their version information, and configuration settings. It also details which components are currently running along with their process ID.

To use this tool, run the following command:

```
/opt/IBM/ITM/bin/cinfo
```

4.3.3 OMEGAMON XE components installed on Windows

The remaining OMEGAMON components were installed onto a single Windows XP machine running in our lab. The first of these components is the Tivoli Enterprise Portal Server (TEPS). The TEPS contains the presentation data that enables users to view workspaces, define situations, and configure historical data collections. As we will be using the monitoring environment to view data produced by the OMEGAMON XE agents on z/OS, CICS on z/OS, and CICS TG on z/OS, we ensure that the appropriate application support data files have been installed onto the TEPS database on this machine.

The other principle component installed on this machine is the Tivoli Data Warehouse (TDW) used for the storage of historical data migrated from monitoring agents. The warehouse database is created by installing and configuring the Warehouse Proxy. The Warehouse Proxy is responsible for coordinating the retrieval of historical data from the monitoring agent on z/OS into the TDW. The other agent we installed to work with the TDW is the

Warehouse Summarization and Pruning Agent. This agent performs two tasks: First, it aggregates the existing data in the TDW into averages over specified periods of time, and second, it deletes data older than a specified age in order to keep the TDW size manageable.

As with the installation on Linux for System z, this is a two-step process where the base ITM components are installed initially (that is, the TEPS, Warehouse Proxy, and Warehouse Summarization and Pruning Agent) followed by the installation of the OMEGAMON XE application support into the TEPS.

Note: The TEPS and TDW require a database product to be installed prior to their installation. We used DB2 UDB V8.2 for this purpose.

We configured the TEPS, Warehouse Proxy, and Warehouse Summarization and Pruning Agent to communicate with the hub TEMS using TCP/IP (IP.PIPE) specifying the host name of the Linux for System z machine (linux1) and the port number on which the hub TEMS has been defined to listen (3456). The process is summarized here:

1. From the Manage Tivoli Enterprise Monitoring Services window, click the application you want to configure and then select **Action** → **Reconfigure....** If the application has not been initially configured, you may need to select **Action** → **Configure....** Figure 4-4 shows the selection of communication protocol for communicating with the hub TEMS.

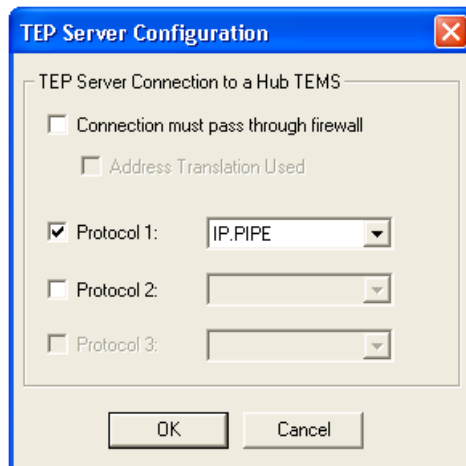


Figure 4-4 Protocol selection for communication with hub TEMS

2. Click **OK**. On the next window, we specified the IP address of the Linux for System z and the port number. Figure 4-5 shows the configuration window. Click **OK**.

The image shows a 'TEP Server Configuration' dialog box with a blue title bar and a close button in the top right. It contains several sections for configuring connections to the TEMS hub:

- IP.UDP Settings of the TEMS:** Hostname or IP Address is 'LEIGHTON', Port number and/or Port Pools is '1918'.
- IP.PPIPE Settings of the TEMS:** Hostname or IP Address is '9.12.4.111', Port number is '3456'.
- IP.SPIPE Settings of the TEMS:** Hostname or IP Address is 'LEIGHTON', Port number is '3660'.
- SNA Settings of the TEMS:** Network Name, LU Name, LU6.2 LOGMODE (set to 'CANCTDCS'), TP Name (set to 'SNASOCKETS' via a dropdown), and Local LU Alias (empty). A note below states '(LU Alias is not required if using default)'.
- Entry Options:** Two radio buttons: 'Use case as typed' (unselected) and 'Convert to upper case' (selected).
- NAT Settings:** A button at the bottom left.
- Buttons:** 'OK' and 'Cancel' buttons at the bottom right.

Figure 4-5 IP.PPIPE connection settings for communicating with the hub TEMS

Table 4-7 shows the communication settings applied for the Tivoli Enterprise Portal Server, Warehouse Proxy, and Summarization and Pruning Agent.

Table 4-7 Connection settings for communicating with the hub TEMS

Setting	Value
Connection protocol 1	IP.PPIPE
Host name or IP Address of TEMS	9.12.4.111
Port number	3456

When started, the TEPS and these agents will make a connection to the hub TEMS.

Tip: It can take over a minute for the TEPS to make a connection to the hub TEMS and complete initialization, during which time no clients can make a connection to the TEPS. The time taken is primarily dependent on the network traffic between the TEPS and hub TEMS. You can reconfigure the TEPS to display its status to a Windows console in order to see quickly and clearly when initialization has completed and also in the event of a communication problem between the TEPS and hub TEMS.

To enable this function, from the Manage Tivoli Enterprise Monitoring Services window, click the Tivoli Enterprise Portal Server application and then select **Actions** → **Change Startup...** In the dialog box that appears, check the box next to **Allow Service to Interact with Desktop** and click **OK**.

Further configuration is needed for the Warehouse Proxy and the Summarization and Pruning Agent. As the TDW resides on the same machine as the Warehouse Proxy, we were able to use all the default values when creating the ODBC data source.

Figure 4-6 shows the Warehouse Proxy configuration we used.

Configure DB2 Data Source for Warehouse Proxy

Data Source Name: ITM Warehouse

Database Name: Warehouse

Please enter your Database Administrator ID and Password below:

Admin User ID: db2admin

Admin Password: [masked]

Please enter the Database User ID and Password required for connecting to the Warehouse Data Source:

Database User ID: ITMUser

Database Password: [masked]

Reenter Password: [masked]

☒ Synchronize TEPS Warehouse Information

OK Cancel

Figure 4-6 Warehouse Proxy configuration

If your TDW resides on a different machine than the Warehouse Proxy, which is an uncommon although not unrecommended approach, then you will need to manually configure the ODBC data source connection using the Windows Administrative Tools found in the Control Panel. This is an advanced topic beyond the scope of this book.

Note: When uninstalling ITM/OMEGAMON from a Windows machine, you may find that the ODBC drivers are not removed. This may cause a problem if you reinstall and try to configure the TEPS or Warehouse Proxy to a different data source. You will have to manually remove the drivers using the Windows Administrative Tools found in the Control Panel.

The final configuration is of the Summarization and Pruning agent. We again used all the default settings. Figure 4-7 on page 139 shows the configuration window.

The screenshot shows a Windows-style dialog box titled "Configure Summarization and Pruning Agent". It has a tabbed interface with tabs for "Sources", "Defaults", "Scheduling", "Work Days", and "Additional Parameters". The "Defaults" tab is selected. The main area contains several configuration fields: "JDBC Drivers" with a text box containing "C:\Program Files\IBM\SQLLIB\java\db2java.zip" and "Add" and "Delete" buttons; "Database" with a dropdown menu set to "DB2"; "Warehouse URL" with a text box containing "jdbc:db2:WAREHOUTS"; "Warehouse Driver" with a text box containing "COM.ibm.db2.jdbc.app.DB2Driver"; "Warehouse User" with a text box containing "ITMUser"; "Warehouse Password" with a text box containing "*****" and a "Test database connection" button below it; "TEP Server Host" with a text box containing "localhost"; and "TEP Server Port" with a text box containing "1920". At the bottom right are "Save", "Reload", and "Close" buttons.

Figure 4-7 Summarization and Pruning agent configuration

From this window, you can define the default summarizations you want to perform on warehoused data and also when to prune older data that is no longer needed (settings specific to certain tables can be made later using the Tivoli Enterprise Portal client). The agent creates a JDBC driver in order to connect to the TDW. It is useful to test the database connection to ensure there are no connection issues. The user name and password should be for the default database user created by the Warehouse Proxy, as shown in Figure 4-6 on

page 138. This is the (non-admin) user ID that all connections use, through the TDW, to request data and perform summarization.

Any changes to the warehouse configuration will be detailed in Chapter 8, “Historical data analysis” on page 293, which deals with scenarios using historical data.

4.3.4 Accessing data using the Tivoli Enterprise Portal client

We accessed the monitoring environment using the Tivoli Enterprise Portal (TEP) client. Both versions of the TEP client were used when running our scenarios.

TEP Desktop client

The desktop client is an installable version of the TEP. As the client is installed onto the user's machine, it gives a better overall performance. However, you need to ensure that all application support has been installed to the client machine; otherwise, workspaces will not be displayed correctly. Also, if maintenance or upgrades are released for any monitored product then it has to be applied to every client machine to remain current.

TEP Browser client

The browser client is an applet downloaded from the TEPS when the client machine attempts to connect. The performance is slower than the desktop client and the initial download of the applet can be quite slow, especially on a low-bandwidth network. It does have the advantage that maintenance and upgrades need only be performed on the TEPS machine, which are then downloaded to any clients when they next connect. Currently, the TEP Browser client supports only the Internet Explorer browser.

Note: As the TEP Browser applet downloaded is very large, you are strongly recommended to reconfigure the Java plug-in settings used by Internet Explorer. From the Advanced settings in the Java Plug-in Control Panel, set the following Java Runtime Parameters:

`-Xms64m -Xmx256m`

This increases the initial heap size of the JVM to 64 MB and a maximum size of 256 MB. You should also adjust the size of cache if needed. We recommend increasing the cache size to an unlimited size, if possible. If not, increase it to at least 100 MB.

4.4 WebSphere Application Server configuration

Another significant part of our scenario is the configuration of WebSphere Application Server. Although the objective of this book is not to explain how WebSphere Application Server works, we will give a brief overview of the basic configuration steps when used in conjunction with the CICS TG.

We installed WebSphere Application Server V6.1 for Linux on System z with the latest available Fix Pack (Fix Pack 11), and in order to not have too complex of an environment, we decided not to activate the WebSphere Security.

Note: The install procedure for WebSphere Application Server requires a graphical environment, therefore you will need to enable a graphical display manager when installing on Linux.

4.4.1 Resource Adapter creation

CICS TG v7.1 on z/OS provides two JCA resource adapters:

cicsecl.rar	For use in a non-transactional environment
cicseclXA.rar	For use in a transactional environments using XA support

The adapters are available in the deployable directory of the CICS TG installation path and they must be transferred to the WebSphere Application Server machine and installed into WebSphere Application Server. In our environment, we used only cicseclXA.rar, as our scenario in Chapter 7, “High availability with XA and OMEGAMON XE” on page 241) required the use of transactional support.

You should perform the following steps to install a Resource Adapter into WebSphere Application Server:

1. Open the Administrative Console and log in. (We used the URL <http://9.12.4.111:9060/ibm/console>.)
2. In the Administrative Console Welcome window, click **Resources** and then **Resource Adapters**.

3. Now click **Install RAR**, which presents you with the Install RAR file window (Figure 4-8), select **Local Path**, and then **Browse** to locate the resource adapter file you have previously transferred to your machine.

Install RAR File

Use this page to install a RAR file in one of two ways. You can either upload a RAR file from the local file system, or specify an existing RAR file on a server. The RAR file must be installed at the node level, and you can select the node below.

Path

☒ Local path:

Specify path

☐ Server path:

Specify path

Scope

Node

linux1Node01

Figure 4-8 RAR file installation

4. Finally, add your changes to the master repository by clicking **Save** and then **Save** again. You should now have a Resource Adapter called ECIRResourceAdapter.

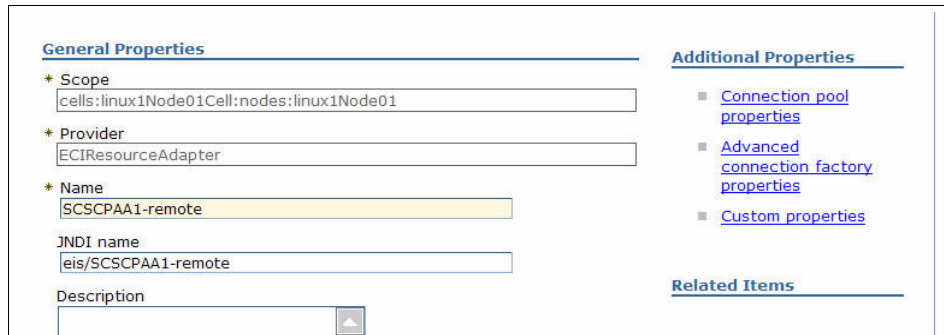
4.4.2 Connection factory creation

The next step is the creation of a connection factory. Each installed resource adapter should have one or more connection factories associated with it. The connection factory is used to define the properties of each connection to a target CICS system. It has to contain the following information:

- ▶ The CICS region that will be the target of the request.
- ▶ The connection details for the CICS TG that will be used.
- ▶ The security credentials that will be used for communication.
- ▶ The CICS mirror transaction ID that should be used.

Note: At least one connection factory is required to use a resource adapter; however, it is also possible to create multiple connection factories each with different options (such as differing mirror transaction IDs), which allows different applications to use connections with different properties, such as security or priority, and also provides a level of isolation since each connection pool is specific to a given connection factory.

5. In the Resource Adapters window, click the **ECIResourceAdapter** just created; the Configuration window is displayed listing the General Properties and the Additional Properties of the resource adapter. Usually there is no need to change the general properties of the Resource Adapter.



The screenshot shows a configuration window for the **ECIResourceAdapter**. It is divided into two main sections: **General Properties** and **Additional Properties**.

General Properties:

- Scope:** cells:linux1Node01Cell:nodes:linux1Node01
- Provider:** ECIResourceAdapter
- Name:** SCSCPAA1-remote (highlighted in yellow)
- JNDI name:** eis/SCSCPAA1-remote
- Description:** (empty field with a small icon to the right)

Additional Properties:

- [Connection pool properties](#)
- [Advanced connection factory properties](#)
- [Custom properties](#)

Related Items: (empty section)

Figure 4-9 Connection factory creation

6. Select the **J2C connection factories** link and then click **New**, which displays the General Properties window of the connection factory to be created.

For our environment, we chose the value *SCSCPAA1-remote* as the connection factory name for our first connection factory (Figure 4-8 on page 142). SCSCPAA1 refers to the name of the CICS region, and remote refers to the topology of CICS TG in use.

We then created an additional Connection Factories for our 2PC scenario environment and named this *SCSCPAA1-XAremote1*.

Note: When you create a Connection Factory for an ECIXAResourceAdapter, in the General Properties window, you will have an additional area named *Authentication alias for XA recovery*. This field is not used by the CICS ECI XA resource adapter, as XA recovery flows do not require credentials.

4.4.3 Connection factory customization

Having created the two connection factories, the final step is to modify the custom properties.

7. Upon clicking **Custom Properties**, the window shown in Figure 4-10 appears.

Name	Value	Description	Required
TraceLevel	1	TraceLevel	false
TPNName		TPNName	false
Password		Password	false
UserName		UserName	false
TranName		TranName	false
ConnectionURL	tcp://9.12.4.15	ConnectionURL	false
ServerName		ServerName	false
ClientSecurity		ClientSecurity	false
KeyRingPassword		KeyRingPassword	false
SocketConnectTimeout	0	SocketConnectTimeout	false
PortNumber	2006	PortNumber	false
KeyRingClass		KeyRingClass	false
CipherSuites		CipherSuites	false
ServerSecurity		ServerSecurity	false
Total 14			

Figure 4-10 Connection factory Custom Properties - SCSCPAA1-XAremote

The custom properties are the CICS specific parameters. For the first connection factory (SCSCPAA1-XAremote), we set the values shown in Figure 4-10 and then saved our configuration.

- ▶ ConnectionURL(tcp://9.12.4.75)
This is the TCP/IP address of the Gateway daemon.
- ▶ ServerName(SCSCPAA1)
This is name of the CICS region as know by the CICS TG. When using EXCI, this is the CICS APPLID. We did not set this value and instead allowed the Gateway daemon to set the default using the DFHJVSYSTEM_00 variable (for more details on defining this variable, refer to Chapter 2, “Configuring the CICS TG” on page 25.
- ▶ PortNumber (2006)
This is the port number that the Gateway daemon is using, which in our case is the default port 2006.

For the second connection factory (*SCSCPAA1-XAremote1*), we set the following values, and then saved our configuration:

- `ConnectionURL(tcp://9.12.4.75)`

This is the TCP/IP address of the Gateway daemon.

- `ServerName`

This is name of the CICS region as known by the CICS TG. When using EXCI, this is the CICS APPLID. We did not set this value and instead allowed the Gateway daemon to set the default using the `DFHJVSYSTEM_00` variable.

- `PortNumber (2007)`

This is the port number that the Gateway daemon is using, which in our case is the new port 2007, which is used in the 2PC scenario where two separate Gateway daemons are required.

Note: In our environment, in order to reduce the tests' complexity, we did not activate security, either in WebSphere Application Server or in CICS. However, one way of activating security is to use the J2C alias mechanism to define a set of security credentials to be used for a specific Connection Factory. For more details on JCA security scenarios, refer to *CICS Transaction Gateway for z/OS Version 6.1*, SG24-7161.

4.4.4 Application deployment

8. The last step to be carried out is the deployment of the J2EE application. Once again, we used the WebSphere Administration console.
 - In the WebSphere Application Server Administrative Console Welcome window, select **Applications** → **Install New Application** and the window in Figure 4-11 is displayed.
 - Enter the location of the EAR file you wish to deploy (we used CTGTesterCCIXA).

Preparing for the application installation

Specify the EAR, WAR, JAR, or SAR module to upload and install.

Path to the new application

☒ Local file system

Full path

☐ Remote file system

Full path

Context root Used only for standalone Web modules (.war files) a

How do you want to install the application?

☐ Prompt me only when additional information is required.

☒ Show me all installation options and parameters.

Figure 4-11 J2EE application deployment

- Remember to click **Show me all installation options and parameters**, which will allow you to update the resource references, and then click **Next**.
- Select the **Use default virtual host name for Web and SIP modules** button and click **Next** again. You are then prompted with the details of the 11 step process for installing new applications.

- You can skip the first 6 steps and go directly to step 7 in order to remap the resource references to the connection factories previously defined.
- The mapping between the resource-reference known by the application and the one you have created at the connection factory level needs to be set. So, check the box next to the application EJB module, browse to the Target Resource JNDI Name, choose the right one, and press **Apply**.

We used the connection factory *SCSCPAA1-XAremote* for both resource references when using our 1PC scenario so that only one resource manager was in use. For our 2PC scenario, we used both connection factories *SCSCPAA1-XAremote* and *SCSCPAA1-XAremote1* representing two recoverable resources (in our CICS regions), as shown in Figure 4-12.

Select	Module	EJB	URI	Resource Reference	Target Resource JNDI Name
<input type="checkbox"/>	CTGTesterCCIXAEJB	CTGTesterCCIXA	CTGTesterCCIXAEJB.jar,META-INF/ejb-jar.xml	ECIAXA	<input type="text" value="eis/SCSCPAA1-XAremote"/> <input type="button" value="Browse..."/>
<input type="checkbox"/>	CTGTesterCCIXAEJB	CTGTesterCCIXA	CTGTesterCCIXAEJB.jar,META-INF/ejb-jar.xml	ECIBXA	<input type="text" value="eis/SCSCPAA1-XAremote"/> <input type="button" value="Browse..."/>

Figure 4-12 Step 7 for 1PC and CICS TG in Port Sharing

- Press **Next** until step 11 and then **Finish**.

To test the application, in the WebSphere Administrative Console Welcome window, select **Applications** → **Enterprise Applications** and start your application. Click the application itself and, under Detail Properties, press **View Deployment Descriptor** in order to check the right context root for it. For our application, it was CTGTesterCCIWebXA, so the correct URL to use was `http://9.12.4.111:9080/CTGTesterCCIWebXA`.

Note: Remember that every time you modify a connection factory value, you will have to close and restart WebSphere Application Server in order for these changes to be reflected in the runtime system.

Remapping a resource-reference

In order to remap the Resource Reference for a deployed WebSphere Application Server application, it is not necessary to uninstall the complete application. Instead, do the following steps:

1. Open the WebSphere Administrative console and select **Applications** → **Enterprise Applications**.
2. Select the application you need to work on by clicking it and, in the window that appears, look at the References area.
3. Click **Resource references** (Figure 4-13).



Figure 4-13 Application resource references

The Resource references page will appear, where you can modify the Target Resource JNDI Names association, as previously shown.

4. Save the changes and stop and restart the application.

4.5 Workload and baseline performance

Once the environment was created, we created and ran various workloads through the Gateway daemons in the CICS regions and measured the response. The purpose of this was twofold: First, it enabled us to verify that the environment was initially functioning as expected, and second, it gave us some baseline performance figures to show the optimal performance of environment prior to introducing problems in the scenarios discussed in the subsequent chapters.

4.5.1 Workload used

Our main workload generation tool is a Java application that enables us to simulate many users concurrently and repeatedly by sending requests to CICS through a Gateway daemon. The number of concurrent connections and the number of iterations each connection makes is fully configurable using a properties file. Each connection is executed in a separate Java thread on the client machine and loops for the amount of iterations or running time requested.

The workload tool creates and issues requests to the Gateway daemon using the base Java API provided by CICS TG. As we are using Gateway daemons on z/OS, all requests are viewed as Extended Call Interface (ECI) requests by the client application. The Gateway daemon converts these requests into an EXCI request to CICS, which is how they are viewed as by the CICS region. Each issued transaction is defined as synchronous with the transaction type *synconreturn*. This means a synchronization point, that is, the point at which the changes made during a request are completed and can be committed, occurs each time a flow of data is made between a Gateway daemon and CICS region. Therefore, each flow or unit of work is viewed as a distinct transaction and we are not using more complex or *extended logical unit of work* (extended LUW) transactions in this exercise.

Our workload tool provided the following configurable parameters:

► Terms

This represents the number of Java client thread instances to generate. Each instance will attempt to open a connection to the targeted Gateway daemon and runs in its own thread of execution. Under normal circumstances this number should be less than the value of the maximum number of Connection Manager threads defined for the Gateway daemon connecting to. If not, the Gateway daemon will not have the capacity to handle the number of incoming connection requests and some client instances will fail with a connection refused.

► Warm-up time

This represents a time period, in seconds, given at the start of execution to allow the Gateway daemon to create the number of Connection Manager and Worker threads in order to adequately cope with the number of connection requests. This value should be considered when measuring the performance of CICS TG, because the creation of new threads requires extra processing by the Gateway daemon.

► Runtime

This represents a time period in seconds to execute the main workload into the Gateway daemon and onto CICS. Each client instance defined using the Terms parameter will be making repeated *synconreturn* requests and waiting for the return from CICS during this time.

► Cool-down time

This represents a time period in seconds given at the end of execution where the client instances make no further requests to the Gateway daemon but will wait to allow any in-flight requests to complete.

- Think time

This represents a time period in seconds that each thread will wait between completing one transaction request and issuing a new request in order to mimic a user response interval a real-life application might experience before issuing a further request. Should this value be set to 0, then no thinking time is allocated, and a new request is issued as soon as the previous one completes.

- COMMAREA size

This represents the size in bytes of the COMMAREA, that is, the data buffer used by a CICS program to hold input and output data when making a request. A larger COMMAREA size will have an effect on performance due to the increase in payload.

- CICS program name

This represents the target program we wish to execute in CICS.

- Gateway daemon address

This represents the address, port number, and protocol used to make a connection to a Gateway daemon. In our basic environment, we have two Gateway daemons both listening on the same port number, so we do not know which CICS TG will ultimately receive the request.

- CICS region name

This represents the jobname of the CICS region to which we want the request to be routed. In our basic environment, as we do not know which Gateway daemon is processing the request and each Gateway daemon has its own default CICS region to which it communicates, we will typically leave this value blank and let the Gateway daemon handle the routing.

In the CICS regions, we installed two simple programs that were executed as part of our workload. These programs are designed to perform basic tasks within CICS to simulate the time taken by more complex operations that would be typically found in a customer's environment. Most of our workload called a COBOL program called ECIPROG, which reads a parameter from the COMMAREA and performs a task based on the command. In most cases, we issue a call for the transaction to wait within CICS before issuing a return back to the Gateway daemon. The basic setting was to cause a delay in CICS for one second. To do this, ECIPROG called a second program called CICDELAY, an assembler program that actually causes the wait. In our scenario testing, we varied the length of the delay in CICS to simulate different behaviors.

For our workload running through WebSphere Application Server, the client application we used was the sample CICSTesterCCIXA enterprise application. Installation and configuration of this application is described in 4.4, "WebSphere

Application Server configuration” on page 141. In order to provide the correct level of the workload, we used a third-party tool designed specifically for creating a heavy load on the system, test functional behavior, and measure performance. The tool provided similar functionality as described above with number of concurrent clients, runtime, and think time parameters.

If you wish to reproduce the tests we describe in the scenarios, you can chose the workload tool with which you are most familiar.

4.5.2 Baseline performance figures

We ran our basic workload generation tool using the following settings and recorded the following set of results as our performance baseline figures (see Table 4-8). The response times for CICS and CICS TG were determined using the time stamps each request is given when it enters and leaves the Gateway daemon. Figures are given for both Gateway daemons we configured (SCST711 and SCST714). Where the workload was very light, the response times were so quick such that the delay could not be detected. As the workload increases, the processing time in both CICS and CICS TG increases.

Table 4-8 Baseline test properties and figures

Value	Test 1	Test 2	Test 3
Maximum Connection Manager threads	100	100	100
Maximum Worker threads	100	100	100
Maximum CICS sessions	100	100	100
CICS MAX TASK	100	100	100
CICS open tasks	50	100	100
Delay in CICS (ms)	0.0	0.0	0.0
Terms	10	100	100
Think time (sec.)	0.1	0.1	0.01
Runtime (sec.)	1200	1200	1200
COMMAREA size (bytes)	100	100	100
CICS program name	CICDELAY	CICDELAY	CICDELAY
Throughput (trans per sec.)	98	961	2475
Total transactions	118000	1154000	2975000
SCSCT711 connections allocated	4	52	48

Value	Test 1	Test 2	Test 3
SCSCT711 CICS response time (sec.)	0.0	0.001	0.025
SCSCT711 CICS TG response time (sec.)	0.0	0.002	0.026
SCSCT714 connections allocated	6	48	52
SCSCT714 CICS response time (sec.)	0.0	0.001	0.0028
SCSCT714 CICS TG response time (sec.)	0.0	0.002	0.029



Diagnosing common problems

This chapter discusses some common problems and their solutions for the CICS TG on z/OS. The majority of the problems dealt within this chapter are due to configuration. A basic level of z/OS skills is assumed.

This chapter is structured to help quickly identify problems. First of all, the chapter is divided into the following major topics:

- ▶ Problems seen on the Gateway daemon or ctgmaster
- ▶ Gateway and ECI error codes returned by the API
- ▶ Java exceptions thrown by the API

Each of these topics is divided into sections. The heading for each section is the daemon behavior or API error that will be seen by the user.

Where there are a number of possible causes for an error code, we discuss problem diagnosis techniques that can help find the cause. The sections are divided into sub-sections. The heading for the sub-sections is the log message or error code that helps to identify the cause of the problem.

If a problem was been addressed in other sections of the book, this chapter refers to the relevant section that discussed the configuration in detail.

5.1 Function used

The tools used most frequently to diagnose problems in this chapter are:

- Gateway daemon logs

The Gateway daemon writes to two log streams:

- An informational log
- An error and warning log

The error log is often the first port of call when diagnosing problems. The logging destinations are controlled through DD card statements on the EXEC CTGBATCH statement used to invoke the Gateway daemon. The information and error log destinations are defined by the STDOUT and STDERR DD cards, respectively. In Example 5-1, we start the CICS TG and send both the informational and error logs to JES.

Example 5-1 Invoke the Gateway daemon with logging to JES

```
//CTG      EXEC PGM=CTGBATCH,  
//          PARM='&LEOPTS.&CTGHOME./bin/ctgstart -noinput '  
//STEPLIB DD DSN=&CTGHLQ..SCTGLOAD,DISP=SHR  
//STDOUT  DD SYSOUT=*  
//STDERR  DD SYSOUT=*  
//STDENV  DD DSN=&CTGUSR.,DISP=SHR
```

- JVM dumps and the Gateway daemon facility to invoke dumps from the administration console.

The JVM writes a number of dumps in the event of an unexpected termination. These include standard z/OS dumps, such as an CEEDUMP, SVC dump, and SnapTrace. They also include Java specific dumps such as Javacore and Heapdump. We make use of the Javacore dump in this chapter. Javacore is output in plain text and was relatively quick and easy to interpret for the problems with which we dealt.

- Gateway daemon statistics

Statistics were introduced in Version 7.0 and expanded in V7.1. We use V7.0 statistics principally for diagnosing thread resource shortage constraints and V7.1 statistics for diagnosing storage issues. Statistics can be interrogated by issuing modify commands against the Gateway daemon job. A full description of the syntax of the z/OS modify commands is included in the CICS TG Information Center. We provide example commands for the statistics we use in each section.

- ▶ OMEGAMON XE

We use the alert feature of the Tivoli Enterprise Portal (TEP) to give administrators advanced warning when problems might occur.

- ▶ The z/OS system log

This is accessed from the 'LOG' option on the SDSF application. We look for errors in the Gateway daemon error log and cross reference them with the system log. In some cases, z/OS will have written extra information to the system log.

- ▶ The **netstat** utility

The **netstat** utility is a network diagnostic utility that is available from both UNIX System Services and from the command-line option from ISPF. We use **netstat** to discover which TCP/IP sockets are in use on a system, and which tasks are allocated to the sockets.

5.2 Gateway daemon and ctgmaster errors

This section contains the following categories:

- ▶ Gateway daemon fails to startup
 - S806: steplib incorrect
 - CTG8659E RRMS resource manager incorrectly specified
 - CTG6525E Unable to start handler for the tcp: protocol
 - CTG6999E Unable to open configuration file
 - Unable to allocate bytes for GC in j9vmem_reserve_memory
 - JVMDUMP006I "java/lang/OutOfMemoryError"
 - CTG6119E This product will not execute on z/OS.e
- ▶ Gateway daemon logs errors at runtime
 - CTG9300E Writing a record to SMF failed
- ▶ Ctgmaster fails to start up
 - CTG8659 Unable to initialize CTGRRMS services 1003

5.2.1 Gateway daemon fails to startup

Look in the Gateway daemon error log for any abend codes that occurred when the Gateway daemon terminated. The following sections represent abend codes or key log entries that identify some common errors.

S806: steplib incorrect

The abend S806 indicates that there has been a failure to load a module.

The Gateway daemon abends with abend code S806 because it is unable to find a module that it requires. You should:

- ▶ Check the syslog for the following message:
`CSV003I REQUESTED MODULE DFHXCPRX NOT FOUND`
- ▶ Ensure that the STEPLIB is set correctly in the STDENV file, using a statement such as the following one:
`EXCI_LOADLIB="CICSTS31.CICS.SDFHEXCI"`
- ▶ Check that the profile of the user ID that runs the Gateway daemon does not set the STEPLIB to point to a different library that might contain conflicting modules.

Note: In CICS TG V6 and V7.0, JNI code is loaded during initialization. This error occurs at Gateway daemon startup, rather than on the first ECI call, which was the case with CICS TG V5.

In CICS TG V7.1, an environment variable called CTG_EXCI_INIT has been introduced that allows EXCI loading to be disabled. If set to YES, the EXCI is loaded. If set to NO, EXCI is not loaded. If you are using IPIC instead of EXCI and want to avoid STEPLIB configuration issues, set CTG_EXCI_INIT to NO. See the CICS Transaction Gateway Information Center for more information. EXCI requests will fail if CTG_EXCI_INIT is set to NO.

CTG6525E Unable to start handler for the tcp: protocol

When CTG6525E is written to the CICS TG error log, the text of the underlying network exception is written within square brackets (see Example 5-2). The exception text gives further information as to why the protocol handler failed to start.

Example 5-2 Example error due to a network configuration problem

```
10/15/07 03:07:03:046 [0] CTG6525E Unable to start handler for the tcp:
protocol [java.net.BindException: EDC5111I Permission denied.]
```

java.net.BindException: EDC5111I Permission denied

This error can occur if you are using port sharing and the CICS TG jobname has not been added to the TCP/IP parameters dataset that controls which jobs can share a specific port. See “Implementing TCP/IP port sharing” on page 67 for details on how to update the z/OS TCP/IP parameters.

You can display the port reservation list with **netstat -o**. Example 5-3 shows a sample display of **netstat** run under UNIX System Services. We can see that port 2002 has been reserved for SCST712 and SCST716.

Example 5-3 Netstat output of port reservation list

```
$ [SC53] /ctg/scsctg71/scstg712/logs: netstat -o
MVS TCP/IP NETSTAT CS V1R8          TCPIP Name: TCPIP          06:35:04
Port# Prot User      Flags    Range      IP Address
-----
00020 TCP  OMVS      DA
00021 TCP  FTPOE1    DABU          9.12.4.146
00021 TCP  FTPMVS1   DABU          9.12.4.145
...
02002 TCP  SCST712   DASW
02002 TCP  SCST716   DASW
...
10000 UDP  OMVS      DAR      10000-11999
```

java.net.BindException: EDC8115I Address already in use

This means that another process is using a port number that the Gateway daemon is configured to use.

In order to discover which port the Gateway daemon is configured to use:

1. Check which protocol is reporting the error in the CTG6525E message. (TCP was reporting the error in Example 5-2 on page 156.)
2. Look for a configuration parameter with the following syntax in the Gateway INI file:

protocol@<prot>.parameters=port=<portnumber>

Where <prot> is either TCP or SSL and <portnumber> is the number that it is configured to listen on.

In order to discover which port numbers are in use, and the processes listening on them:

1. Start an OMVS shell.
2. Issue **netstat | grep <portnumber>**, where <portnumber> is the same one mentioned previously.

Example 5-4 shows that a job called SCST712 is listening on port 2002.

Example 5-4 Finding out which process is using a specific port

```
$ [SC53] /u/cicsrs3: netstat | grep 2002
SCST712 000099AC 0.0.0.0..2002          0.0.0.0..0          Listen
$ [SC53] /u/cicsrs3:
```

CTG6999E Unable to open configuration file

Part of the Gateway daemon configuration is contained in a z/OS Distributed File Service zSeries File System (zFS) file. CTG6999E indicates that the Gateway daemon was unable to open the file and therefore fails to start.

An example message is shown in Example 5-5. The exception text in square brackets gives the path of the file and the reason that the load failed. The exception text can help you diagnose why the failure occurred.

In Example 5-5, the name of the file we are trying to load is /ctg/scsctg71/scstg712/ctg.ini and the reason is due to a permissions problem.

Example 5-5 Unable to open ini file due to incorrect permissions

```
10/17/07 09:03:35:735 [0] CTG6999E Unable to open configuration file
[java.io.FileNotFoundException:/ctg/scsctg71/scstg712/ctg.ini (EDC5111I
Permission denied.)]
```

EDC5111I Permission denied

The user ID that invokes the Gateway daemon must have read access to the file documented in the exception text. It must also have read and execute access on the directory containing the file.

Further details on the permission problem are written to the Gateway daemon job log (see Example 5-6 on page 159).

Example 5-6 Starting Gateway daemon with incorrect permissions

```
IEF695I START CITGR1G WITH JOBNAME CITGR1G IS ASSIGNED TO USER
CITGR1G
, GROUP CITG
$HASP373 CITGR1G STARTED
$HASP100 BPXAS ON STCINRDR
$HASP373 BPXAS STARTED
ICH408I USER(CITGR1G ) GROUP(CITG ) NAME(CTG GATEWAY DAEMON U) 324
/cicstg/ctg610/config/CITGR1G.ini
CL(DIRSRCH ) FID(01D6C5F0F0F0F100010400002DFB0000)
INSUFFICIENT AUTHORITY TO OPEN
ACCESS INTENT(--X) ACCESS ALLOWED(OTHER ---)
EFFECTIVE UID(0000000013) EFFECTIVE GID(0000007100)
$HASP395 CITGR1G ENDED
```

To resolve the problem, set execute permission on the directories for the Gateway daemon group using the ISHELL command, specifying the relevant directory, and select **Directory** → **Attributes** → **Edit** → **Owning Group**.

Section 2.3.2, “Setting directory permissions” on page 37 gives details on how to set up the correct file permissions.

Unable to allocate bytes for GC in j9vmem_reserve_memory

Example 5-7 shows a CICS TG startup with very little REGION storage, which has lead to an error.

Example 5-7 CICS TG startup with very little REGION storage

```
Error: unable to allocate 134217728 bytes for GC in
j9vmem_reserve_memory.
JVMJ9VM015W Initialization error for library j9gc23(2): Failed to
instantiate h
Could not create the Java virtual machine.
```

This error occurs because the REGION size is very low. There is insufficient storage to start the JVM. The solution is to increase the region size significantly.

The amount of storage available to a task can be controlled a parameter in the JCL called REGION. It can also be controlled by a user exit called IEFUSI. We only discuss the REGION parameter in this chapter. See *z/OS V1R6.0 MVS Installation Exits*, SA22-7593-11 for details on IEFUSI.

The REGION parameter controls the maximum amount of storage a program is allowed to allocate. This is set within the invoking JCL. It can be set either as a parameter on the EXEC call, which invokes the CICS TG, or on the jobcard statement.

We found on our systems that we needed 160 MB to enable the Gateway daemon to start and at least 250 MB to perform a basic workload. Tuning the REGION size parameter is discussed in “JVMDUMP006I java/lang/OutOfMemoryError” on page 160.

The soft and hard limits for storage available to the Gateway daemon process are output to the informational log on startup by message CTG0813I. Example 5-8 shows the values output when the REGION size is 500 MB. (CTG0813I is only output if the debug DD card CTGDBG is active (see “CTGDBG” on page 74).)

Example 5-8 CTGBATCH reporting the storage available to a task

```
CTG0813I CTGBATCH RLIMIT_AS reports current=507M, system max=2047M
```

JVMDUMP006I java/lang/OutOfMemoryError

The reason that the dump occurred is because of the error /java/lang/OutOfMemoryError. This error means that the REGION size was sufficient for the JVM to start, but insufficient for the Gateway daemon to start. Example 5-9 shows an example of the log output in this case.

Example 5-9 OutOfMemoryError message caused by insufficient region size

```
10/08/07 14:13:10:348 Y0" CTG6502I Initial ConnectionManagers = 500,
Maximum ConnectionManagers = 500
10/08/07 14:13:10:349 Y0" CTG6526I Initial Workers = 100, Maximum
Workers = 100
CTG6122E The CICS Transaction Gateway could not execute the command
JVMDUMP006I Processing Dump Event "systhrow", detail
"java/lang/OutOfMemoryError" - Please Wait.
JVMDUMP007I JVM Requesting Snap Dump using
'/ctg/scsctg71/scstg714/logs/Snap0001.20071008.181311.67308946.trc'
JVMDUMP010I Snap Dump written to
/ctg/scsctg71/scstg714/logs/Snap0001.20071008.181311.67308946.trc
JVMDUMP007I JVM Requesting Heap Dump using
'/ctg/scsctg71/scstg714/logs/heapdump.20071008.181311.67308946.phd'
JVMDUMP010I Heap Dump written to
/ctg/scsctg71/scstg714/logs/heapdump.20071008.181311.67308946.phd
JVMDUMP007I JVM Requesting Java Dump using
'/ctg/scsctg71/scstg714/logs/javacore.20071008.181311.67308946.txt'
```

JVMDUMP010I Java Dump written to
/ctg/scsctg71/scstg714/logs/javacore.20071008.181311.67308946.txt

In addition to the logs, the contents of the javacore dump show that it was driven by a memory shortage. Javacore dumps are text files that can be viewed with a text editor. Example 5-10 shows the first few lines of a javacore that was driven by a lack of memory.

Example 5-10 Contents of a javacore driven for a memory shortage

OSECTION	TITLE subcomponent dump routine
NULL	=====
1TISIGINFO	Dump Event "systhrow" (00040000) Detail
	"java/lang/OutOfMemoryError": "Failed to fork OS thread" received
1TIDATETIME	Date: 2007/10/16 at 17:11:20
1TIFILENAME	Javacore filename:
	/ctg/scsctg71/scstg712/logs/javacore.20071016.171119.17761239.txt
NULL	-----

The javacore also shows that the error occurred when we were trying to start another thread (Failed to fork OS thread).

The number of Worker and Connection Manager threads has a large impact on the storage used by the Gateway daemon. To resolve the out of memory issues, you can:

- ▶ Reduce the number of Worker and Connection Manager threads
- ▶ Increase the region size

Tip: The *v7.1 CICS Transaction Gateway Information Center* gives a rule of thumb for calculating the amount of storage needed for the CICS Transaction Gateway. This information is contained in *Threading Considerations* under *Diagnosing Common Problems* in *Problems with the Gateway daemon*.

Tip: Statistics have been introduced in CICS TG V7.1 that display the amount of storage available to the Gateway daemon and the amount of storage it is currently using. When sufficient REGION storage has been allocated to allow the Gateway daemon to start up and run a typical workload, you can use these statistics to see how much storage is required and tune the REGION size.

- ▶ SE_SELIM is equal to the amount of REGION storage specified in bytes unless REGION has been overridden by the IEFUSI exit.
- ▶ SE_CELOAL is the amount of storage currently in use in bytes. (This value can go up or down.)

CTG6119E This product will not execute on z/OS.e

If you are running a full version of z/OS, this message may indicate that the temporary directory is full.

The usual behavior of the Gateway daemon startup script **ctgstart**, when TMPDIR is full, is to fail to start with a CTG6120E error code. However, the point at which the failure occurs can vary and the behavior of **ctgstart** is undefined when the temporary directory is full. Other error messages such as CTG6119E may occur.

The temporary directory is referenced by the TMPDIR environment variable. Set TMPDIR to a file system that has free space and restart the Gateway daemon.

Tip: We used separate file systems for the temporary directory and for logging and tracing in our setup to prevent this error occurring. See 2.3.1, “Defining an HFS” on page 33 for further details.

5.2.2 Gateway daemon log errors at runtime

Here we discuss Gateway daemon log errors at runtime.

CTG9300E Writing a record to SMF failed

An example message is shown here:

```
10/23/07 11:20:00:348 [1] CTG9300E Writing a record to SMF failed with  
return code (Errno='157', Errno2='9210408')
```

This message indicates that the Gateway daemon has failed to write a record to SMF. It is a general purpose log that covers a range of failure conditions. One such condition is when the active SMF datasets fill up because they are not off-loaded to archive files frequently enough.

If this message appears, show the status of SMF by using the following display command:

```
/D SMF
```

Example 5-11 shows the output of a DISPLAY SMF command when the Gateway daemon has reported a CTG9300E.

Example 5-11 Output of a DISPLAY SMF command

RESPONSE=SC66						
IEE974I 11.13.10 SMF DATA SETS 207						
	NAME	VOLSER	SIZE(BLKS)	%FULL	STATUS	
	P-SYS1.SC66.MAN1	PAG660	3000	100	DUMP REQUIRED	
	S-SYS1.SC66.MAN2	PAG660	3000	100	DUMP REQUIRED	
	S-SYS1.SC66.MAN3	PAG660	3000	100	DUMP REQUIRED	

To diagnose the problem further, look in the system log for SMF error messages. The following messages (see Example 5-12) indicate that the SMF buffer is filling up.

Example 5-12 Errors in the system log due to the SMF buffer becoming full

```
*IEE986E SMF HAS USED    93% OF AVAILABLE BUFFER SPACE
*IEE986E SMF HAS USED   100% OF AVAILABLE BUFFER SPACE
*IEE979W SMF DATA LOST - NO BUFFER SPACE AVAILABLE TIME=11:16:25
```

5.2.3 Ctgmater fails to start up

Look in ctgmater's error log. The section headings below refer to key log entries that identify common problems.

CTG8659E Unable to initialize CTGRRMS services 1003

This can indicate that a dataset required by CTGRRMS is not included in the MVS LNKLIST. Look in the system error log for messages written at the same time stamp as CTG8659E.

If the LNKLIST is incorrectly specified, a message indicating that CTGINIT can not be found is written to the system log at the time of the failure, as shown in Example 5-13.

Example 5-13 System log errors when ctginit is not found

```
BPXM023I (CTGUSER) CTG8930I SCST715 CTGMASTER INITIALIZATION IS
STARTING
CSV003I REQUESTED MODULE CTGINIT NOT FOUND
CSV028I ABEND806-04 JOBNAME=MSTRJCL STEPNAME=IHV
IEA989I SLIP TRAP ID=X806 MATCHED. JOBNAME=CTGRRMS , ASID=0082.
IEF170I 1 CTGRRMS CSV003I REQUESTED MODULE CTGINIT NOT FOUND
IEE824I CTGRRMS FAILED, TERMINATED
BPXM023I (CTGUSER) CTG8700E SCST715 CTGMASTER FAILED TO INITIALIZE
Jobname Procstep Stepname CPU Time EXCPs RC
SCST715 STARTING MASTER 00:00:00 247 00
IEF404I SCST715 - ENDED - TIME=07.23.44 - ASID=0075 - SC53
$HASP395 SCST715 ENDED
```

SCTGLINK is the dataset installed with the CICS TG that contains CTGINIT. Ctgmaster requires that SCTGLINK is APF authorized and included in the MVS LNKLIST. See “Enabling CTGRRMS services” on page 48 for details.

Tip: This message is written to the system log when a ctgmaster startup is initialized:

```
BPXM023I (CTGUSER) CTG8930I SCST715 CTGMASTER INITIALIZATION IS
STARTING
```

This message is written to the system log if the startup is unsuccessful:

```
BPXM023I (CTGUSER) CTG8700E <JOBNAME> CTGMASTER FAILED TO INITIALIZE
```

This message is written if the startup is successful:

```
CTG8931I SCST715 CTGMASTER INITIALIZATION IS COMPLETE. RESOURCE
MANAGER NAME=<MASTER RRMNAME>
```

<JOBNAME> is the name of the CTGMASTER job. <MASTER RRMNAME> is the name of the resource manager the master registers as under RRMS.

If the operating system has written information as to why the startup fails, it will be between the CTG8930 and CTG8931 messages.

Tip: You can verify which libraries are included in the LNKLIST by issuing a DISPLAY command. Issue the command /D PROG,LNKLST against the system log and check the output for the SCTGLINK dataset.

5.3 Gateway and ECI return codes

This section contains the following categories:

- ▶ ECI_ERR_NO_CICS -3
- ▶ ECI_ERR_RESOURCE_SHORTAGE -16
- ▶ ECI_ERR_SYSTEM_ERROR -9
- ▶ ECI_ERR_SECURITY -27
- ▶ ERROR_WORK_WAS_REFUSED 61445 (0xF005)

The Java API can return both Gateway and ECI return codes. As a general rule, ECI return codes indicate an error within CICS or communicating to CICS. Gateway return codes generally indicate that an error occurred in the Gateway daemon processing, rather than in communicating to CICS.

Use the *getRc* method on the ECIRequest object in the client application to get the return code. GetRc() returns a Gateway return code if one is set. If no Gateway return code has been set, it returns an ECI return code. If no error has occurred, it returns 0.

Gateway return codes are in the range 0xF000 to 0xF011. ECI return codes are in the range from -1 to -1001.

This chapter deals with a subset of the return codes that illustrate common error scenarios. Each heading below represents a return code. See the *CICS Transaction Gateway Information Center* for a full list of return codes.

Tip: The return codes are documented in the Javadoc™ information section of the *CICS Transaction Gateway Information Center*. The ECIReturnCodes and GatewayReturnCodes class descriptions contain descriptions of the return codes.

To view a list of all the return codes possible and their numerical values, click an individual return code in either ECIReturnCodes or GatewayReturnCodes, then click **ConstantFieldValues**. The subsequent window lists all the values for all types of error code.

5.3.1 ECI_ERR_NO_CICS -3

Verify that the CICS region is active. If the CICS region is active, there may be a configuration problem preventing communication between the Gateway daemon and CICS.

Figure 5-1 shows an inactive CICS region. In this figure, SCSCPAA1 and SCSCPAA4 are active and region SCSCPTA2 is inactive. This is indicated by SCSCPTA2 being shown as greyed out.



Figure 5-1 TEP view of CICS down

ECI_ERR_NO_CICS does not usually result in any entries being written to the error log. When no entries are written to the error log, we need to switch on JNI trace to debug this error. “JNI trace” on page 81 explains how to active JNI trace.

Look for a message CCL6822E in the resultant trace (see Example 5-14). We are looking for an error with Reason code 203, which is NO_CICS. We then need to look for the contents of *subreason field-1* in the message. Each heading within this section is a subreason code.

Example 5-14 Error log entry for ECI_ERR_NO_CICS

```
08:59:00.990 [050f0395,149d844000000265,Worker-99  ] : [0] CTG6822E
EXCI function error.  Function Call = 3, Response = 8, Reason = 203,
Subreason field-1 = 0x5c, subreason field-2 = 0x00, Cics_Rc = -3.
```

Tip: For full details on EXCI return and reason codes, see the CICS Transaction Server V3.2 Information Center or the *CICS External Interfaces Guide*, SC34-6830.

Reason code 203 NO_CICS is documented as meaning that:

- ▶ The target CICS system is not available.
- ▶ The target CICS has not yet opened IRC.
- ▶ The target connection is out of service.
- ▶ The relevant EXCI connection definition is not installed in the target CICS.

Subreason field-1= X'005C'

This means IRERRNLG "System not logged on".

One reason for this message is that IRC is not open. Check to see if IRC is open by using the CICS command CEMT INQ IRC. See 2.4.7, "Opening interregion communication" on page 56 for an example of this command.

If the CEMT INQUIRE shows IRC as active, then check that the CICS region user ID has READ access to the RACF FACILITY class profile DFHAPPL.<CICS_applid>.

Subreason field-1 = X'0068'

This means IRERRNSS "Secondary system not in primary LCB".

This error occurs because the value specified for the CICS TG DFHJVPIPE environment variable does not correspond with the NETNAME value of any installed CICS CONNECTION definition.

Check that DFHJVPIPE has been set correctly and that a CONNECTION definition with a NETNAME value, which is the same as the DFHJVPIPE value, has been installed on the CICS region.

Subreason Field-1 = X'007C'

This means RERRIQP. Primary is in QUIESCE.

This error can occur if the connection has been set out of service in CICS. Check that there is an in-service connection using the CICS command CEMT I CONN NET(NETNAME), where NETNAME is the netname defined in DFHJVPIPE.

Figure 5-2 gives an example of an in-service connection. The same information can be seen through the TEP Client (see Figure 5-4 on page 175).

```
, I CONN NET(SCSCT714)
,RESULT - OVERTYPE TO MODIFY
,Connection(T714)
,Netname(SCSCT714)
,Pendstatus(, )
,Servstatus(Inservice,)
,Connstatus(, )
,Accessmethod(Irc)
,Protocol(Exci)
,Purgetype(, )
,XlInstatus()
,Recovstatus(, )
,Uowaction(, )
,Cqp(Notsupported)
,Grname()
,Membername()
,Affinity(, )
,Nqname()
,Remotesystem()
,+,Rname()
,SYSID=PAA1,APPLID=SCSCPA1
,TIME: 07.55.56 DATE: 10.16.07
,PF,1,HELP,2,HEX,3,END, ,5,VAR, ,7,SBH,8,SFH, ,10,SB,11,SF,
```

Figure 5-2 An in-service connection

See 2.4.1, “Creating a CONNECTION definition” on page 52 for details on creating the definitions in CICS. See 2.3.4, “Creating an STDENV file” on page 40 for details on configuring the matching definition on the CICS TG.

Tip: The TEP for OMEGAMON XE warns the systems administrator with a *CICSTG_Health_Critical* alert if a significant number of requests are failing with ECI_ERR_NO_CICS or ECI_ERR_RESOURCE_SHORTAGE. This allows the problem to be addressed as soon as it occurs, rather than waiting for users to report errors.

CICSTG_Health_Critical is triggered if the GD_CHEALTH statistic reaches 0. In CICS TG V7.0, requests fail for 60 seconds before the health statistic reaches 0. A configuration parameter has been introduced in CICS TG V7.1 to allow the user to reduce this time.

A health statistic of 0 occurs if all requests within a set time period fail with ECI_ERR_NO_CICS or ECI_ERR_RESOURCE_SHORTAGE. The default interval is 60 seconds. Set the healthinterval parameter in the Gateway INI file to the required time period.

5.3.2 ECI_ERR_RESOURCE_SHORTAGE -16

This return code indicates that there are not enough resources to complete a request. A common cause of this error is a shortage of EXCI pipes. Look in the Gateway daemon error log for EXCI related errors at the same time this request was issued.

CTG6992E No pipe available

This indicates that a request has failed because the receivecount limit on the EXCI session in the connection between the Gateway daemon and CICS Region is exceeded (see Example 5-15).

Example 5-15 Error due to EXCI pipe resource constraint

```
10/10/07 04:35:41:088 [0] CTG6882E No pipe available. An attempt was
made to open a pipe to server {SERVERNAME}, but no free receive
sessions were available. CTG currently holds NUM
pipes.
```

You can see the receive count value for a CICS region and the number of open pipes in OMEGAMON XE.

CTG6876E EXCI error

The full message text is given in Example 5-16. Look at the EXCI Reason and subreason fields to further diagnose the error.

Example 5-16 JINI trace point due to EXCI problem

```
04:39:09.218 [050f098a,149993100000021a,Worker-25  ] : [5] CTG6876E
EXCI error.  Function Call = 2, Response = 16, EXCI Reason = 608,
Subreason field-1 = 0x38, subreason field-2 = 0x80000064, ctg_rc=-16.
```

EXCI Reason = 608,Subreason field-1 = 0x38

This indicates that there was a failure to allocate a pipe due to a resource shortage of some kind. There are a variety of reasons that this message can occur. One possibility is the logon limit being exceeded.

Consideration: Once the logonlimit has been exceeded for the Gateway daemon's address space, the subreason field-1=0x38 error will be produced, regardless of whether the CICS region the requests are being set to is active.

EXCI pipe limits

Two limits that are commonly reached are:

- ▶ The EXCI logon limit
An MVS address space is allowed 100 EXCI pipes by default. APAR PQ92943 provided the ability to increase this up to a maximum of 250 using a CICS subsystem initialization parameter called LOGONLIM.
- ▶ The EXCI session receive count
The CICS session definition related to the connection being used for the EXCI connection with the Gateway daemon sets a limit on the total number of pipes. (The receive count can be seen on the TEP Client window in Figure 5-4 on page 175.)

The value set for LOGONLIM is reported in several ways. The following message in the Gateway daemon's informational log reports the LOGONLIM as the number of pipes available to the Gateway:

```
CTG6898I <LOGONLIM> EXCI pipes are available for use by the CICS
Transaction Gateway.
```

The CS_LOGONLIM statistic and EXCI Pipe Limit in the CICS TS Regions Summary pane on the TEP provide the same information.

Attention: The receive count value needs to be larger than the logon limit, in order to deal with queued pipe allocations in CICS. If you set the receive count to the same value as the logon limit, errors will occur when the maximum pipe limit is approached. The receivecount parameter must be configured to be a value of two or more higher than LOGONLIM.

The symptoms of this problem are a sequence of CTG6882E messages where the number of pipes reported in use starts off one less than the logon limit and reduces by 1 in each subsequent message. For example, if LOGONLIM and RECEIVECOUNT are both set to 100, errors are reported with 99 pipes, 98 pipes, 97 pipes, and so on. Example 5-17 shows a log of this sequence.

Example 5-17 The receive count set at the same value as the logon limit

```
10/10/07 04:35:40:594 [0] CTG6882E No pipe available. An attempt was
made to open a pipe to server {SCSCT711} but no free receive sessions
were available. CTG currently holds 99 pipes.
10/10/07 04:35:41:088 [0] CTG6882E No pipe available. An attempt was
made to open a pipe to server {SCSCT711} but no free receive sessions
were available. CTG currently holds 98 pipes.
10/10/07 04:35:41:151 [0] CTG6882E No pipe available. An attempt was
made to open a pipe to server {SCSCT711} but no free receive sessions
were available. CTG currently holds 97 pipes.
```

Understanding the Gateway daemon pipe requirements

In order to increase performance, the Gateway daemon caches and re-uses EXCI pipes once they are allocated. It is not possible to share EXCI pipes between Worker threads, so caching takes place on a per Worker thread basis.

The maximum number of pipes that the Gateway daemon will allocate depends on:

- ▶ The value specified for CTG_PIPE_REUSE
- ▶ The maximum number of Worker threads
- ▶ The number of different CICS regions the Gateway sends requests to

The maximum number of pipes used is summarized in Table 5-1.

Table 5-1 Maximum EXCI pipes used

CTG_PIPE_REUSE	Maximum pipes used
ALL	Maximum number of Worker threads multiplied by the number CICS regions
ONE	Maximum number of Worker threads

All the CICS regions used by the Gateway can be listed by viewing the CS_LLIST stat. Issue the following z/OS modify command against the Gateway job to see that stat:

```
/F SCSCT711,APPL=STATS,GS=CS_LLIST
```

The number of CICS regions returned by CS_LLIST should be used in place of the number of CICS regions in Table 5-1.

The maximum number of Worker threads allowed within a Gateway daemon address space is controlled by maxworker in the Gateway daemon configuration file. This value is written to the output log in message CTG6526I.

Configuring the Gateway daemon to resolve the problem

There are a number of ways the configuration can be modified to resolve a pipe shortage:

- Increase the number of pipes available to the CICS TG.
Increase the receive count in the CICS sessions definition. If more than 100 pipes are required per address space, increase LOGONLIM in the SYS1.PARMLIB. When running CICS V2, you need to apply PQ92943 to allow the LOGONLIM to be increased.

Attention: When the Gateway daemon and CICS region are on different LPARs, each pipe uses a slot on the default cross-system coupling facility (XCF) group DFHIR000. The total number of slots within an XCF group is limited to 2047. XCF is used by other applications in addition to the CICS TG/EXCI. Running the Gateway daemon and CICS regions on different LPARs is not recommended to avoid excessive use of XCF slots.

There is also a hardcoded limit of 8,192 EXCI pipes per LPAR, which applies when Gateway daemons and CICS regions are on the same LPAR.

Consult your systems administrator before increasing the number of EXCI pipes used by the Gateway daemon.

- Reduce the number of Worker threads.

Section 6.3.2, “Gateway daemon Worker thread constraint” on page 213 shows a formula that describes how to calculate the number of Worker threads required for a specific workload and how to identify Worker thread shortages.

- If multiple servers are being used, select CTG_PIPE_REUSE=ONE.

If more than one CICS region is being used, we recommend you set CTG_PIPE_REUSE to ONE. This will set an upper limit on the number of pipes that can be allocated to the same as the maximum number of Worker threads. There is a possible performance impact associated with this option due to reallocation of pipes.

Tip: The CICS TG V7.1 statistic CSX_IAVRESP is the average response time for CICS transactions. It includes pipe allocation and deallocation times in addition to the CICS transaction request itself. You can estimate the implications on response time of changing the pipe allocation model by running the same, steady workload with pipe allocation set to ONE and ALL and comparing the results for CSX_IAVRESP.

To get the results for this statistic, substitute the server you are interested in for X. For example, to get information about server SCSCPJA7, you would issue the following command:

```
/F SC SCT717,APPL=STATS,GS=CSSCSCPJA7_IIVRESP
```

Tip: The CS_IREALLOC statistic can be used as an indication of the efficiency of the pipe caching model when CTG_PIPE_REUSE=ONE. This count is incremented each time the Worker thread needs to deallocate a pipe and reallocate a new one, instead of using the pipe that it has cached. Comparing CS_IREALLOC to the total number of requests in an interval CS_IALLREQ gives an indication of the “hit rate” for pipe allocation.

Diagnosing EXCI pipe shortages using the TEP Client

The TEP Client gives details that give a better indication of the cause of the error. Figure 5-3 is an example of where the logonlimit was exceeded.

The screenshot shows two windows from the TEP Client. The top window, titled 'CICS TS Regions Summary', contains a table with the following data:

System ID	Gateway Daemon Name	EXCI Pipes Allocated	CICS TS Region Count	Requests Executed	EXCI Pipe Limit	EXCI Pipe Reallocations	EXCI Pipe Allocation Failures
SC53	SCSCT712	100	3	24610	100	0	4

The bottom window, titled 'Individual CICS TS Region Summaries', contains a table with the following data:

System ID	CICS TS Region Jobname	CICS TS Region Applid	EXCI Pipes Allocated	Requests Executed	Requests Executed Per Minute	EXCI Pipe Allocate Failures	EXCI Pipe Failures
SC53	SCSCPTA2	SCSCPTA2	100	24606	0	0	
SC53	SCSCT716	SCSCT716	0	3	0	3	
SC53	SCSCPJA6	SCSCPJA6	0	1	0	1	

Figure 5-3 CICS inactive and logonlimit exceeded

To diagnose the failure, look at the CICS TS regions view. Check if any pipe allocation failures have occurred in the EXCI pipe allocation failures field. To understand why the pipe failures occurred, look at the distribution of EXCI pipes across CICS regions in the Individual CICS TS regions summary. If CICS region jobname is blank, it indicates that the job for the CICS region is inactive. If the jobname field exists, the CICS region is likely to be active. If pipe allocation failures are occurring in an active region, we can assume that it is due to an allocation limit being exceeded.

Check the logon limit and the total number of pipes allocated using the EXCI pipes limit and EXCI pipes allocated fields in the summary view. If the pipes allocated is equal to the limit, then it may be due to a logon limit being exceeded.

Another possibility is that the receivecount for the connection has been exceeded. To obtain the receivecount value, go to the CICS region where the failures are occurring and select the **Connections analysis** view. The number of links defined is the same as receivecount. If EXCI pipes allocated is equal to the Number of links defined, it is likely that the receive count is being exceeded. In Figure 5-4 on page 175, connection TI12 has a receive count of 200.

Connections Analysis							
	Connection Name	Connected System VTAM Applid	Connected System Name	Connection Type	Connection Status	Number of Links Defined	Number of Links in Use
	PAA1	SCSCPAA1	Unknown	IRC_XM	ACQ_INS	198	0
	PAA4	SCSCPAA4	Unknown	IRC_XM	ACQ_INS	198	0
	PJA6	SCSCPJA6	SCSCPJA6	IRC_XM	REL_INS	198	0
	PJA7	SCSCPJA7	Unknown	IRC_XM	ACQ_INS	198	0
	T712	SCSCT712	Unknown	IRC_EXCI	REL_INS	200	100
	T716	SCSCT716	Unknown	IRC_EXCI	REL_INS	200	0

Figure 5-4 Connections Analysis

The TEP Client makes it possible to see the distribution of pipes across Gateway daemons. For example, in Figure 5-3 on page 174, all of the pipes are allocated to SCSCPTA2 and none to any of the other regions.

5.3.3 ECI_ERR_SYSTEM_ERROR -9

System error is returned for a range of error conditions. Look for errors in the Gateway daemon log at the same time as the application reported the error. The headings in this section are log entries that are driven by some common error scenarios.

CTG6823E EXCI DPL_REQUEST specific error

An example of a full message is given in Example 5-18. The RESP and RESP2 fields give further information about the error.

Example 5-18 Error for an unexpected abend in the mirror transaction

```
10/18/07 09:52:11:676 [0] CTG6823E EXCI DPL_REQUEST specific error.
RESP value= 0x51, RESP2 value = 0x00, Abend Code = Rc = -9.
```

CTG6823E contains response codes from a failed EXCI DPL request. These errors represent return codes from CICS that are equivalent to RESP and RESP2 values in an EXEC CICS LINK command. The *CICS Application Programming Reference*, SC34-6434 provides details on EXEC CICS LINK.

Tip: The return code field being traced is dpl_request from DFHXCIS. For a more detailed explanation, see the DPL_Fields topic in Chapter 9, “The EXCI CALL interface”, of the *CICS Transaction Server for z/OS External Interfaces Guide Version 3 Release 2*, SC34-6830.

TERMERR Resp value 0x51 (81)

TERMERR is documented in *CICS Application Programming Reference*, SC34-6434 as “an irrecoverable error occurs during the conversation with the mirror (for example, if the session fails, or if the server region fails)”.

Some examples of situations that could drive this error are:

- ▶ The Gateway daemon issues a request to CICS region A. The program is defined as remote and runs on a different CICS region B.
- ▶ While the transaction is in progress, the mirror transaction it is running under is force-purged on region B.
- ▶ Region B terminates unexpectedly while the transaction is in progress.

CTG6818E Begin Context failed

Begin Context calls are involved when units of work span multiple ECI programs. CTG6818E messages can be driven by failures in extended logical units of work or XA transactions.

Example 5-19 shows a Begin Context failure that occurred due to an incorrect RRM name.

Example 5-19 Begin Context failure due to incorrect RRM name

CTG6818E Begin Context failed. RRM Return code = 0X756

This message is often an indication that the Gateway daemon failed to register correctly with Resource Recovery Management Services (RRMS). The Gateway daemon uses the RRMS subsystem to manage transactional work. It registers with RRMS on startup. Look for messages written to the error log at startup time.

A CTG6806E message indicates a RRMS registration failure. The return code gives information about the reason for the failure. The Name parameter is the name the Gateway daemon attempted to register under. This is controlled by the CTG_RRMNAME parameter in the STDENV file.

Example 5-20 Failure to register with RRMS

CTG6806E Register with RRS failed. Return code = 0X300. Name = "SCSCT712".

The following list gives some common return codes:

► 300 - RM_NAME_INVALID

This error indicates that the name chosen for the resource manager is not valid. The name specified by CTG_RRMNAME must end with .IBM.UA when support for XA transactions is not enabled, or when support for XA transactions is enabled with TCP/IP load balancing of XA transactions.

A missing .IBM.UA is a common cause of this error. Section 2.3.4, “Creating an STDENV file” on page 40 describes how to set the CTG_RRMNAME parameter and explains the rules for naming it in detail.

► 700 - RM_NAME_REGISTERED

This error indicates that another Gateway daemon has already registered with the same name. All Gateways must have a unique name.

5.3.4 ECI_ERR_SECURITY_ERROR -27

There are several steps required in order for the Gateway daemon to be able to authenticate successfully. The setup we used is described in 2.2.2, “Basic security considerations” on page 30.

A call can fail with a security error for the following reasons:

- Surrogate checking has been enabled in the EXCI options table DFHXCOPT.
- RACF program control must be active for the CICS Transaction Gateway load library SCTGLOAD, and the CICS Transaction Server on z/OS SDFHEXCI load library.
- Extended attributes settings are incorrect for certain HFS files.
- The Gateway daemon is configured to check user IDs and passwords with RACF and an invalid password is supplied on a request.

These problems are explained in detail in the ECI_ERR_SECURITY_ERROR return code topic in the CICS TG V7.1 Information Center.

5.3.5 Invalid data returned in the COMMAREA

ASCII to EBCDIC conversion issues can cause the data to be invalid.

When a Java application is running on a distributed platform and communicating with a Gateway daemon on z/OS, either:

- ▶ Recompile DFHCNV to convert the COMMAREA to the codepage that the distributed platform requires.
- ▶ Perform a code-page conversion within the calling Java application.

A sample program called *Ecib2*, which is shipped with the CICS TG can be configured to perform a code page conversion on a COMMAREA. As a quick way to check whether the problem is due to an incorrect codepage conversion, run Ecib2 with the ASCII and EBCDIC options and see which version of the data is correct.

5.3.6 ERROR_WORK_WAS_REFUSED 61445 (0xF005)

This indicates that the Gateway daemon was not able to allocate a Worker thread in the specified timeout. A Worker thread is allocated for the duration of a transaction, as shown in Figure 5-5.

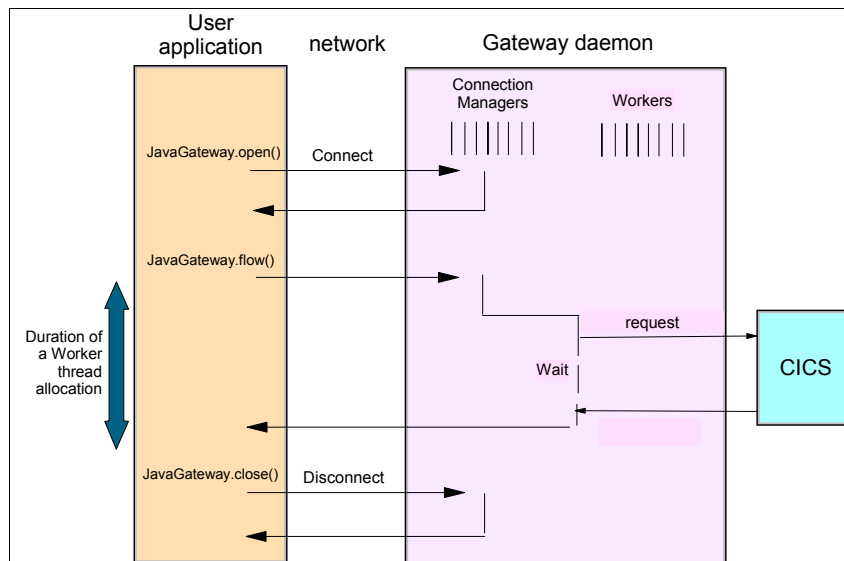


Figure 5-5 Worker thread usage

The maximum number of Worker threads and the timeout are controlled by `maxworker` and `workertimeout` in the `.ini` file. Several scenarios in this book provide walkthroughs of example Worker thread shortages and advise how to diagnose and resolve the problem.

There are other Gateway return codes (11, in fact). The main one, `ERROR_CONNECTION_FAILED`, which could be caused by a lack of CMs, is covered in 5.4.5, “`IOException CTG6668E`” on page 188.

Tip: The following functions can be very useful in diagnosing thread shortage problems and tuning the system to use the correct resources for the expected workload:

- ▶ Average response time statistics (introduced in CICS TG V7.1)
- ▶ Historical data written to SMF (introduced in CICS TG V7.1)
- ▶ The charts and alerts provided by the TEP
- ▶ Historical data available from OMEGAMON XE

See 6.3.2, “Gateway daemon Worker thread constraint” on page 213 and 8.3, “Using historical data to diagnose system problems” on page 297 for walkthroughs on using these tools to diagnose common scenarios.

5.4 Base Java API and J2EE exceptions

In addition to return codes, the Java API can throw exceptions. The base API will generally throw an `IOException` in the event of a network error.

5.4.1 Java application hangs when trying to connect

A Java application hangs when attempting to connect to the Gateway daemon. This can occur if the Gateway daemon has run out of storage when dynamically creating a Connection Manager thread.

Check the Gateway daemon error log to see if any dumps have been issued. The presence of entries in the Gateway daemon error log starting with *JVMDUMP* indicates that the JVM has issued a dump.

If the JVMDUMP messages state that we are processing a `java/lang/OutOfMemoryError`, this implies a memory shortage. Example 5-21 shows what the log entries look like for a memory shortage.

The most likely cause of a memory error is that we are creating too many Connection manager or Worker threads. This can be confirmed by looking at the contents of the javacore text file produced. (The JVMDUMP entries give the location of the file.) If the problem was caused by a thread issue, the core file will contain entries like the one shown in Example 5-21.

Example 5-21 Sample of output from a memory shortage javacore

```

NULL
-----
-
OSECTION      TITLE subcomponent dump routine
NULL          =====
1TISIGINFO    Dump Event "systhrow" (00040000) Detail
"java/lang/OutOfMemoryError":"Failed to fork OS thread" received
1TIDATETIME   Date:                2007/10/17 at 08:53:50
1TIFILENAME   Javacore filename:
/ctg/scsctg71/scstg712/logs/javacore.20071017.085349.17761353.txt

NULL
-----
-
OSECTION      THREADS subcomponent dump routine
NULL          =====
NULL
1XMCURTHDINFO Current Thread Details
NULL          -----
3XMTHREADINFO "tcp" (TID:0x1EF5E300, sys_thread_t:0x1E6368AC,
state:R, native ID:0x147A45D0) prio=5
4XESTACKTRACE at java/lang/Thread.startImpl(Native Method)
4XESTACKTRACE at java/lang/Thread.start(Thread.java:970)
4XESTACKTRACE at
com/ibm/ctg/server/ThreadManager.createObject(Bytecode PC:127)
4XESTACKTRACE at
com/ibm/ctg/server/ThreadManager.allocate(Bytecode PC:204)
4XESTACKTRACE at
com/ibm/ctg/server/ManagedResources.allocateConnectionManager(Bytecode
PC:19)
4XESTACKTRACE at com/ibm/ctg/server/SocketHandler.run(Bytecode
PC:319)
4XESTACKTRACE at java/lang/Thread.run(Thread.java:801)

```


NULL

The dump title shows that there was not enough storage to fork a thread and the stack back trace shows that we were attempting to allocate a Connection Manager thread at this point.

Important: To reduce the chances of storage errors occurring at runtime, set the minimum number of Connection managers and Worker threads to the maximum. This makes the CICS TG create all the threads at startup. If the REGION size is too small for the number of threads required, a memory dump will be produced at startup. The REGION size can then be adjusted to allow the CICS TG to start correctly.

See “JVMDUMP006I java/lang/OutOfMemoryError” on page 160 for details on how to configure REGION size correctly.

In the above memory shortage, despite the Gateway daemon being hung, administration commands still function. In order to reconfigure the Gateway to resolve the problem, it will need to be restarted. A controlled shutdown can be issued, by issuing the shutdown z/OS modify command or by purging the job:

```
/F SCSC712,APPL=SHUT
```

This is preferable to forcing the Gateway daemon to abort by cancelling the Job.

Tip: It is still possible to issue administrative commands to the Gateway daemon when it is not responding. This is because the administration logic runs under different threads of execution than the main Gateway daemon code.

This can make it possible to get extra debug information out of the CICS TG process when diagnosing a hang. To do so, issue z/OS modify commands to the Gateway daemon. Some examples of commands that would give useful information to IBM service representatives are:

To produce a javacore dump, which will give information about all currently executing threads, run:

```
/F SCSCT712,APPL=DUMP,JVM
```

If the zFS does not contain enough file space for a javacore, a stack back trace of all executing threads can be sent to the Gateway daemon informational log by running:

```
/F SCSCT712,APPL=DUMP,JVMSTACK
```

To send details about the Gateway daemon configuration to the Gateway daemon informational log, issue the following z/OS modify command:

```
/F SCSCT712,APPL=DUMP,CTGINFO
```

Using OMEGAMON XE alerts for first failure data capture

This section contains an example situation where an OMEGAMON XE alert can be set up to provide first failure data capture information. In our example, we want a dump to be taken automatically if a Gateway daemon hangs.

We never expect the Gateway daemon to be idle for more than one hour. We set up a situation that would occur if no requests run through the Gateway daemon on four successive 15 minute intervals. The alert will drive a modify command to ask the Gateway daemon to produce a dump. We do not want multiple dumps to be produced if the Gateway stays hung. The following steps illustrate how this could be achieved through the TEP GUI:

1. Select the CICS TG instance to which the alert applies in the physical explorer pane.
2. Right-click the instance and select **Situations**.
3. Click the **Create Situation** icon in the top left hand corner, as shown in Figure 5-6 on page 183. This results in a Create Situation pop-up menu.

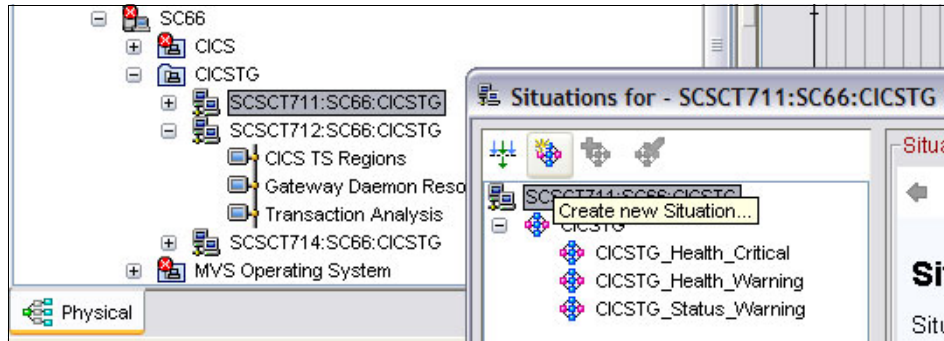


Figure 5-6 Create situation

4. Enter a name and description for the situation. (We chose CICSTG_HangFFDC.)

Attention: If the OK button is greyed out, check that you have not included restricted characters, for example, spaces, in the name.

5. Click **OK** to open the Select Condition pop-up menu. Select the attribute that is being used to trigger the alert. Choose the **CICS TG CICS TS region details** group and select the **Number of requests executed per minute** field within the group.

6. Click **OK**. In the formula tab of the Situation Editor, set the trigger condition to be requests per minute = 0. Leave the sampling interval as the default of 15 minutes (see Figure 5-7).

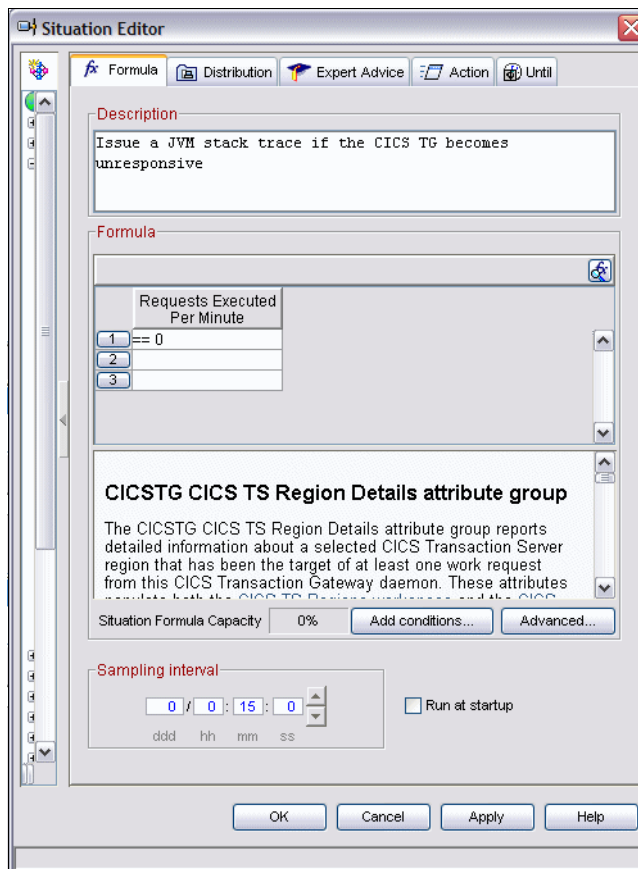


Figure 5-7 The situation editor formula tab

7. Click **Advanced** and set Situation Persistence to four consecutive samples (see Figure 5-8 on page 185). This means that the situation will only be triggered if the condition is true on four consecutive tests. Each test will take place on the sampling interval.

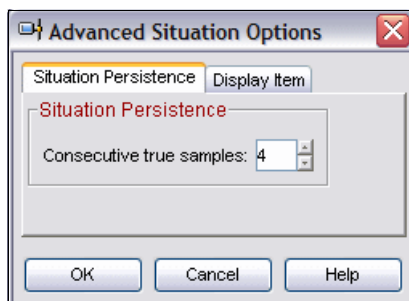


Figure 5-8 Situation persistence pop-up menu

8. Go back to the Situation Editor and select the **Actions** tab. Enter the CICS TG dump option that you want to be taken in the System Command field. In this example, we set it to dump a stack trace of each Java thread to the Gateway daemon informational log. To prevent multiple dumps being issued for the same hang, select the **Don't take action twice in a row (wait until situation goes false and then true again)** radio button.

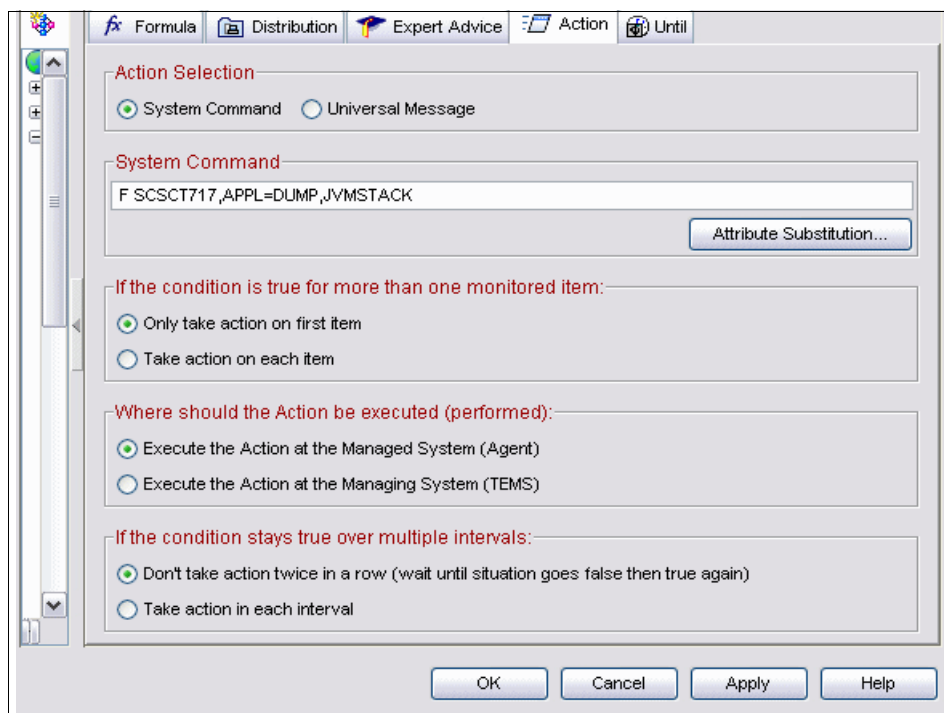


Figure 5-9 Action command to issue a dump

9. To test the condition, we left the Gateway daemon idle for one hour and verified that Java stack information was written to the informational log.

5.4.2 IOException CTG6651E

The full text of the exception is Unable to connect to the Gateway daemon (see Example 5-22). The network address it failed to connect to and the Java exception detailing the failure are in square brackets. This text gives useful information for diagnosing the problem.

Example 5-22 Unable to connect

```
java.io.IOException: CTG6651E Unable to connect to the Gateway daemon.  
[address= wtsc66.itso.ibm.com, port = 2002] [java.net.ConnectException:  
Connection refused: connect]
```

java.net.ConnectException: Connection refused: connect

This may indicate that the port number is incorrectly specified in the URL given to the application or that the Gateway daemon is not running on the target machine.

To see if the Gateway daemon is running, view the status of the Gateway daemon job under SDSF on the target machine. If the job is running, use the **netstat** command to check what port on which it is listening. Section 2.5.3, “TCP/IP netstat command” on page 60 gives details on using the **netstat** command from z/OS and Section 2.3.3, “Creating a Gateway daemon configuration file” on page 38 describes how to configure the port number on which the Gateway daemon listens.

java.net.UnknownHostException

This may indicate that the address specified in the URL given to the application is correct. To verify the URL, issue the operating system **ping** command against the network address given in the exception text.

5.4.3 javax.resource.ResourceException

If an XA request fails and the transaction is rolled back, view the WebSphere SystemOut log for exceptions that occurred at the time of the error. The following sections give entries in the log that can help diagnose some common problems.

CTG9681E not enabled to support XA

The full text of the exception is Caused by: javax.resource.ResourceException: CTG9681E Connected to a CICS TG that is not enabled to support XA transactions.

This error occurs if xasupport=on has not been configured in the Gateway daemon configuration file. See 2.3.3, “Creating a Gateway daemon configuration file” on page 38 for details on this parameter and how to alter it.

CTG9638E Transaction Abend

The full text of the exception is CTG9638E Transaction Abend occurred in CICS. Abend Code=: , error code:.

This message is given if the CICS transaction abended. In some cases, the abend code may not be propagated back to the J2EE code and the Abend Code field is blank. In this instance, look in the CICS logs for a transaction abend occurring at the same time as CTG9638E was written to the WebSphere log (see Example 5-23).

Example 5-23 CICS log of a transaction abend

```
DFHAC2236 10/19/2007 10:13:58 SCSCPAA1 Transaction CSMI abend ARXC in
program CICDELAY resources will be backed out. EXCI job =
SCSCT711.SCSCT711. - SC66.
```

A couple of common abends are summarized here:

AEIO

The program could not be found on CICS. Either the program name is incorrect or the CICS region has not been correctly set up.

ARXC

An attempt was made to run an XA transaction or an extended logical unit of work. RRMS=NO has been specified in the CICS Region's SIT.

These return codes are documented in the CICS Transaction CHLP. CHLP can be run from a CICS terminal. A full set of CICS abend codes can be found in *CICS Transaction Server on z/OS Messages and Codes*, GC34-6827.

5.4.4 javax.transaction.xa.XAException

The text of the exception is CTG9613E An XAException occurred processing XA request.

If an XA request fails and the transaction is rolled back, view the WebSphere SystemOut log for exceptions that occurred at the time of the error. The headings in this section represent errors in the WebSphere log.

WTRN0078E

The full text of the message is WTRN0078E:An attempt by the transaction manager to call start on a transactional resource has resulted in an error. The error code was XAER_RMERR. The exception stack trace follows:.

error text is error:XAER_RMERR

XAERR_RMERR can occur for many different circumstances. Look for a previous error in the WebSphere log, such as CTG9681 (not enabled to support XA).

XAERR_RMERR can be driven by the CICS TG JNI code. If no earlier entries in the WebSphere log appear related to the problem, look in the Gateway daemon error log for entries at the same time as the XAERR_RMERR was logged in WebSphere.

5.4.5 IOException CTG6668E

The full exception text is CTG6668E Initial handshake flow failed. The message contains text in square brackets that gives further details on the reason for the exception. Use this text to diagnose the problem further. Example 5-24 is a sample message.

Example 5-24 Connect failed due to insufficient Connection managers

CTG6668E Initial handshake flow failed. [ERROR_CONNECTION_FAILED]

ERROR_CONNECTION_FAILED

This may indicate that the Gateway daemon has insufficient Connection Manager threads to process incoming application connections. Every concurrent TCP/IP connection from an application to the Gateway daemon requires a Connection Manager thread.

To confirm whether failures to allocate Connection Manager threads are occurring, check the CM_LTIMEOUTS statistic. This is incremented each time that a Gateway daemon fails to allocate a Connection manager within a specified period of time. To view the statistic, issue the following modify command against the Gateway daemon job:

```
/F SCSC712,APPL=STATS,GS=CM_LTIMEOUTS
```

The result is sent to the user console (Example 5-25).

Example 5-25 The statistic indicating failed Connection manager allocate attempts

```
RESPONSE=SC53
BPXM023I (CTGUSER)
CTG8239I Response received from CICS Transaction Gateway
CM - Connection manager
      CM_LTIMEOUTS=5 (Number of times connect timeout reached)
```

Tip: OMEGAMON XE ships a sample alert to warn when CM_LTIMEOUTS is being hit. It is called CICSTG_ConnTimeout_Warning and is triggered if 100 timeouts occur within a minute. The trigger can be modified to whatever number of timeouts per minute is relevant to your system setup.

The best way to calculate the number of Connection Manager threads required depends on whether you are using a J2EE application or a base Java application. With J2EE applications, the number of Connection Manager threads depends on the J2C Connection Factory settings in the Resource Adapter. To alter this setting, select **Resources** → **Resource Adapters** → **J2C Connection Factories** on the Integrated Solutions Console. Select the connection factory that is being used by your application and then click **Connection Pool Properties**. Figure 5-10 on page 190 shows an example of some connection pool settings.

WebSphere Application Server maintains a pool of open TCP/IP connections to the CICS TG. The number of connections will be between Minimum Connections and Maximum Connections, as shown in Figure 5-10 on page 190. The number of connections at any time depends on the number of concurrent transactions executing on WebSphere Application Server and how quickly it is configured to close inactive connections.

To avoid problems when WebSphere Application Server is running under high load, maxconnect in the Gateway daemon initialization file must be greater than or equal to Maximum Connections in the connection pool. If you are using multiple WebSphere Application Server instances or multiple resource adapters, maxconnect must be greater than or equal to the sum of the connections for all of the Connection factories.

J2C connection factories > **SCSCPAA1-remote-XA** > **Connection pools**

Use this page to set properties that impact the timing of connection management tasks, which can affect the performance of your application. Consider the default values carefully; your application requirements might warrant changing these values.

Configuration

General Properties	Additional Properties
Scope <input type="text" value="cells:linux1Node01Cell:nodes:linux1Node01"/>	<input type="checkbox"/> Advanced connection pool properties
Connection timeout <input type="text" value="10"/> seconds	<input type="checkbox"/> Connection pool custom properties
Maximum connections <input type="text" value="500"/> connections	
Minimum connections <input type="text" value="50"/> connections	
Reap time <input type="text" value="60"/> seconds	
Unused timeout <input type="text" value="60"/> seconds	
Aged timeout <input type="text" value="120"/> seconds	
Purge policy <input type="text" value="FailingConnectionOnly"/>	
<input type="button" value="Apply"/> <input type="button" value="OK"/> <input type="button" value="Reset"/> <input type="button" value="Cancel"/>	

Figure 5-10 Example connection factory settings

For applications written with the base Java API, the number of connections in use depends on the design of the user application. The general sequence of events for an application is to open a connection using `JavaGateway.open()`, issue any number of requests through `JavaGateway.flow()`, and close the connection using `JavaGateway.close()`. Each instance of a `JavaGateway` object maintains one TCP/IP socket between an open and a close (Figure 5-11 on page 191). The number of Connection managers must be greater than or equal to the concurrent number of `JavaGateway` objects that have been opened by the user's applications.

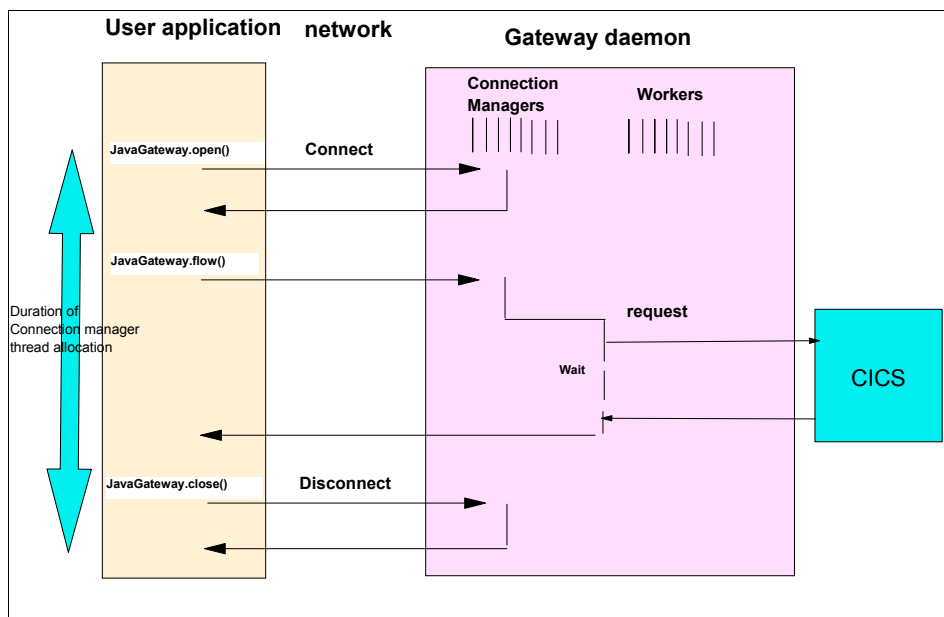



Figure 5-11 Duration of a Connection Manager thread allocation

It is possible to configure the Gateway daemon so that an application will wait for a Connection manager to become free. The `connecttimeout` parameter in the Gateway daemon .ini file controls this value. Reducing the Connection Manager threads available in the pool and relying on this timeout should only be considered if applications open and close connections very frequently.

Details on setting values in the Gateway daemon configuration file can be found in 2.3.3, "Creating a Gateway daemon configuration file" on page 38.



Diagnosing system slow downs

This chapter describes how to diagnose system slow downs that can potentially occur when using the CICS Transaction Gateway on z/OS. We provide several examples of scenarios where both configuration or application programming issues cause a slow down. We show how to identify the problems using a variety of methods, including CICS TG statistics, CICS TG monitoring exits, OMEGAMON XE for CICS TG, and a simple **ping** tool we developed. For each situation, we show how to rectify the underlying issues where appropriate.

6.1 Scenario introduction

The setup is based on the high availability configuration described in 4.2.1, “CICS TG configuration” on page 124, in order to simulate a production environment. Two cloned Gateway daemons were used together with two cloned CICS regions (Figure 6-1).

Figure 6-1 presents a simplified view of the full environment by breaking it into four major components: front end, network, Gateway daemons, and CICS regions. We will use this environment as a basis for our discussion in this chapter.

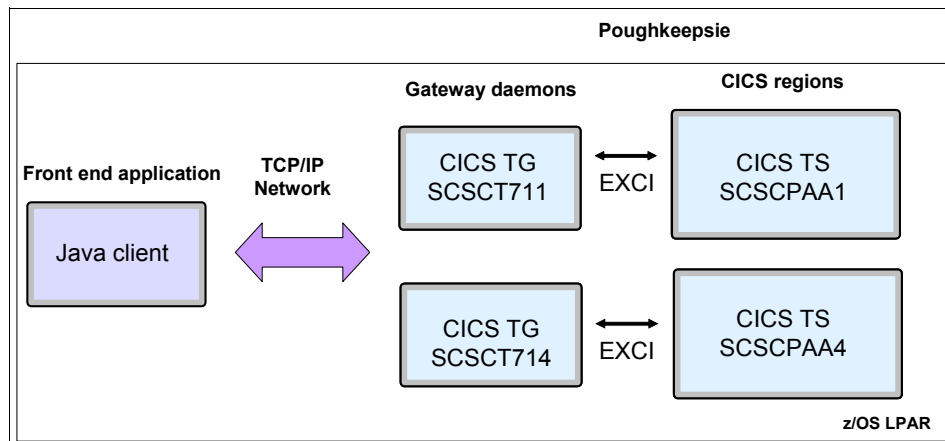


Figure 6-1 Simplified system overview

We configured each Gateway daemon to send requests to a specific CICS region by default. No CICS server was specified in the front-end application, so all requests were routed to the default CICS region. Restricting a Gateway daemon to one CICS region only is the most efficient model in terms of EXCI pipe allocation. Each CICS region listened to on the same TCP/IP port for incoming requests. TCP/IP port sharing was configured in order to distribute incoming application requests across the two Gateway daemons.

The front-end application was written to simulate a large number of concurrent users. It was standalone (it did not need to run under WebSphere) and typically ran on the same LPAR as the Gateway daemon.

The application allowed multiple concurrent connections to be opened to the gateway. The number of connections could be configured. Each connection ran in a separate thread of execution on the front end and issued transactions in a loop. It was possible to configure the think time between each transaction. The

application used the base Java API and issued synchronous transactions. The transactions were “sync on return” rather than extended logical units of work. This means that a sync point is taken for every flow between the CICS TG and CICS so each flow is a separate transaction that is individually committed.

We used the base setup described in Chapter 4, “Scenario environment” on page 121. We varied the following parameters for individual scenarios within this chapter:

- ▶ The number of concurrent connections to the Gateway daemon
- ▶ The think time between transactions being issued on each connection
- ▶ The response time of the back-end CICS transaction

We document the values we used at the start of each scenario.

The configuration was modified as follows to introduce delays in each of the major components:

1. Network slowdown

The CICS TG and CICS were situated in Poughkeepsie, New York, USA. The front-end application was moved to Hursley, UK, introducing a high network latency.

2. Gateway daemon slowdown

The number of concurrent requests invoked against each Gateway daemon was set to significantly exceed the number of Worker threads available.

Full debug tracing was activated on a Gateway daemon.

3. CICS

The program was coded to have a small delay and the program definition was quasi-reentrant (this means it is not thread safe). A large number of concurrent requests were invoked against this transaction causing queuing, which introduced large delays.

The above delays affected all transactions running through the system. In order to demonstrate how to diagnose more complex issues where a delay is intermittent, we had a scenario where only a minority of requests were delayed.

4. The delay only affects a minority of the requests.

One program is coded to have a delay of 140 ms. A second program is coded to have a delay of 10 seconds. A workload is set up where the majority of requests issue the program with a short delay and a couple of requests issue the program with the large delay.

6.2 Functions used

We used some of the features of the following tools to illustrate how these problems could be resolved:

- ▶ OMEGAMON XE for CICS TG on z/OS
- ▶ The **ctgping** tool
- ▶ Gateway daemon statistics
- ▶ CICS TG monitoring exits

6.2.1 OMEGAMON XE for CICS TG on z/OS

The Tivoli Enterprise Portal (TEP) provides a GUI front end to OMEGAMON XE for CICS TG. OMEGAMON XE tolerates Version 7.1, but does not currently exploit statistics that are new with that release. This means that it will work with Version 7.1 but will only display the statistics that were available on Version 7.0.

Important: You must apply APAR OA2681 to get OMEGAMON toleration for CICS TG V7.1.

Operating system statistics, such as CPU time and I/O counts for each Gateway daemon address space, are also provided by OMEGAMON.

Statistics for the CICS regions used by the CICS TG are accessible from the same workspace. The default workspace charts the values of key statistics. The workspace is fully customizable, allowing all the available statistics to be charted in many different ways.

You can also configure alerts to be triggered on specific conditions so that the system operator is informed immediately of significant events, such as resource shortages. You can also configure actions for OMEGAMON to take on an alert, such as sending a z/OS modify command to the task.

In this chapter, we used the statistics presented in the default CICS TG workspace. We activated a number of existing sample alerts and also created new alerts in order to provide advance warnings of the type of error scenarios we encountered.

Summary of the statistics used in this chapter

- ▶ Gateway daemon Overview → CPU Utilization
- ▶ Gateway daemon Overview → I/O Count
- ▶ Gateway daemon Overview → Requests per minute

- ▶ Gateway daemon Resources → Connection Manager Threads allocated
- ▶ Gateway daemon Resources → Connection Manager threads used
- ▶ Gateway daemon Resources → Connection Manager Number waiting
- ▶ Gateway daemon Resources → Worker threads allocated
- ▶ Gateway daemon Resources → Worker threads used

6.2.2 ctgping

The **ctgping** SupportPac (Example 6-1) is a sample test tool that has been developed to help diagnose CICS TG response problems. It opens connections to a specified Gateway daemon, executes a number of requests, and times the results. The code for **ctgping** is distributed in SupportPac CH50, which is available from the CICS SupportPac Web site:

<http://www.ibm.com/support/docview.wss?rs=1083&uid=swg27007241>

Example 6-1 Sample ctgping output

```
Pinging gateway url: tcp://9.12.4.75:2007
Reply from tcp://9.12.4.75:2007 open=210ms, request=120,100,100,100,90ms, close=101ms
Reply from tcp://9.12.4.75:2007 open=190ms, request=100,100,100,100,100ms, close=91ms
Reply from tcp://9.12.4.75:2007 open=190ms, request=100,100,100,100,100ms, close=91ms
Reply from tcp://9.12.4.75:2007 open=200ms, request=90,90,90,100,100ms, close=101ms
Reply from tcp://9.12.4.75:2007 open=190ms, request=90,100,100,100,190ms, close=100ms

----ctgping statistics----
Opens issued=5,min=190ms, max=210ms, avg=196ms, errors=0
Requests issued=25,min=90ms, max=190ms, avg=102.8ms, errors=0
Closes issued=5,min=91ms, max=101ms, avg=96.8ms, errors=0
```

Open is the time to open a connection between **ctgping** and the Gateway daemon (**ctgping** calls the `open()` method on `JavaGateway` in the base API to do this). Opening a connection involves the Gateway daemon allocating a Connection Manager, creating a TCP/IP socket and handshake flows. Network connections are managed by Connection Manager threads. If open times are slow, it may indicate, for example, a shortage of free Connection Manager threads, a slow network, or a high CPU load.

Close is the time to close a connection between **ctgping** and the Gateway daemon (**ctgping** calls the `close()` method on the `JavaGateway` class to do this). This involves sending a close request to the Gateway daemon, receiving an acknowledgement, and closing the TCP/IP socket. A slow close may indicate a slow network, or a high CPU load on the Gateway daemon.

The **ctgping** tool issues ECI ListSystem requests by default. ListSystems involves creating a Worker thread and making a call to the JNI. (JNI stands for Java Native Interface. When a CICS transaction is invoked, the JNI component issues the EXCI requests.) ListSystems is processed within the Gateway daemon JNI code itself, but does not involve making a request to CICS. Therefore, the ListSystems code path through the Gateway daemon is similar to a CICS transaction's path. A slow request time may indicate a shortage of Worker threads or it may indicate a slow down within the Gateway daemon.

If the -c parameter is specified with **ctgping**, a transaction is invoked on the CICS server. Comparing the request times when running the transaction in CICS and when issuing purely Gateway daemon requests can give an indication as to whether a delay is within the Gateway daemon component or in CICS.

Figure 6-2 illustrates how the timings returned by **ctgping** relate to the gateway daemon threading model.

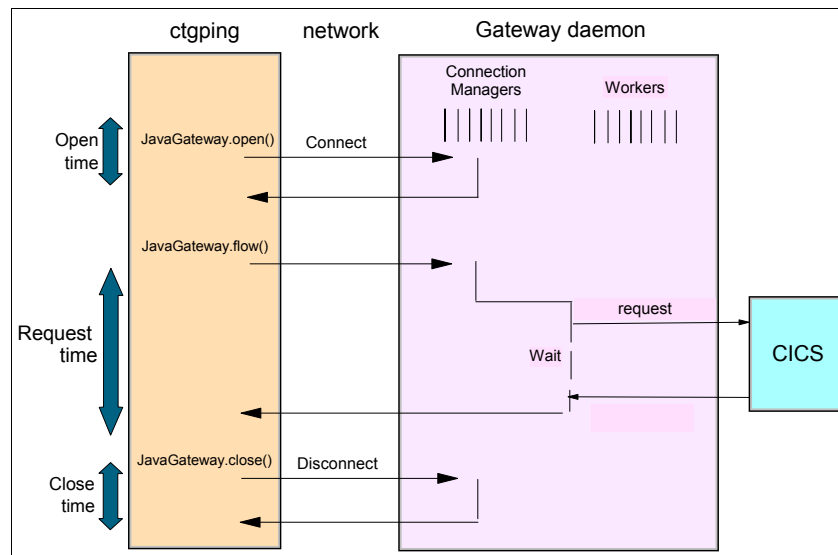


Figure 6-2 *Ctgping measurements*

6.2.3 Gateway daemon statistics

In Version 7.0, statistics were introduced into the CICS TG. They give information about workload, available resources, configuration, and error conditions, which are key to understanding the status and operation of the Gateway daemon available at runtime.

Statistics are grouped into *resource groups* according to the component of the Gateway they represent. The resource groups are:

Connection Manager

Statistics related to the thread pool that handles TCP/IP connections to the Gateway.

CICS Server (all)

This group represent an agglomeration of requests and resources for all the servers being used by the Gateway.

CICS Server (instance)

There is an instance of this group for each CICS server used by the Gateway daemon. The EXCI and IPIC communication protocols are represented here.

Gateway daemon

The core of the Gateway daemon process.

System Environment

Operating system resources relevant to the Gateway daemon, such as Java heap sizes and garbage collection counts.

Protocol handler

Configuration information about the TCP and SSL protocols that are used to connect client applications to the Gateway daemon.

Worker thread

The pool of threads responsible for handling the execution of transactions on CICS server.

On z/OS, these values can be accessed by issuing modify commands against the Gateway daemon job and programmatically through an API. On distributed Gateways, the **ctgadmin** command gives access to the statistics.

Statistics have been considerably enhanced in Version 7.1. The interval statistics mechanism and writing to SMF have been introduced. The total number of individual statistics has been greatly increased. See the *Statistics Introduction* topic in the CICS Transaction Gateway V7.1 documentation for further details.

Tip: Statistics have been added to the Client daemon component of the CICS TG V7.1. This chapter deals with the CICS TG on z/OS. Because the Client daemon is a component of the distributed Gateway only, we do not document Client daemon statistics here.

CICS TG statistics categories

Statistics are categorized as interval, lifetime, startup, and current. Interval statistics are counts or running averages that are reset at the end of a pre-configured interval. Lifetime statistics are counts or running averages, which persist for the lifetime of the gateway daemon. Many statistics have both interval and lifetime equivalents. Current statistics represent transient values that can go up or down over time, such as the number of allocated threads.

The naming convention for statistic IDs is <resource group>_<statistics type><statistics suffix>. A statistics type of I indicates interval, L lifetime, and C current.

Tip: For example, GD_IAVRESP is a statistics ID of AVRESP in the GD (Gateway daemon) resource group and is an interval category of statistic. GD_LAVRESP is the lifetime equivalent of the same statistic.

Statistics related to delays

Average response times can be useful for diagnosing in which component a slow down is occurring, provided that:

- ▶ The majority of transactions are running slowly. (The average response times are for all requests.)
- ▶ The slow down occurs for a significant periods of time. (It must occur for at least the length of one interval for the average response statistic to provide useful results.)

Statistics related to Gateway daemon resource shortages

The CICS TG maintains a pool of Connection Manager and Worker threads. For each connected client (such as a connection in the WebSphere Application Server connection pool), one Connection Manager thread is used in the Gateway daemon, and is held until the client closes the connection or the Gateway times out the connection. In order for an ECI call to be performed through an allocated Connection Manager thread, a thread must be allocated from the Worker thread pool for the duration of the ECI request.

Statistics are available to monitor the number of these threads currently allocated, the number of requests waiting for a thread to become free from the pool, and the number of times that a request times out waiting for a thread to become available.

Summary of the statistics used in this chapter

Table 6-1 summarizes the statistics used in this chapter.

Table 6-1 A summary of the statistics used.

Statistic Name	Description	Available in OMEGAMON XE
GD_IAVRESP and GD_LAVREP	The average time taken (in milliseconds) for the Gateway daemon to respond to all Java client requests. All request types are included.	No
CSX_IAVRESP, CSX_LAVRESP	An indication of the average time taken (in milliseconds) for a connected CICS server X to process a request. For EXCI, only ECI requests are included.	No
WT_ETIMEOUTS	The number of times the Gateway daemon failed to allocate a Worker thread to a Connection Manager within the defined (workertimeout) length of time.	Yes
WT_CALLOC	The current number of Worker threads that are being used by Connection Managers.	Yes
CM_CALLOC	The current number of Connection Manager threads allocated to Java clients.	Yes
CM_CWAITING	The current number of Connection Managers waiting for a Worker thread to become available.	Yes

The descriptions have been taken from the monitoring section in the CICS TG V7.1 Information Center. Refer to the Information Center for details on the entire range of statistics available to the CICS TG.

In general terms, a high value for CS_IAVRESP indicates that there is a delay in CICS. A small value for CSX_IAVRESP but a large value for GD_IAVRESP indicates a delay in the Gateway daemon. A small value for GD_IAVRESP but a slow user application response time indicates a delay in the network or front-end application.

6.2.4 CICS TG monitoring exits

A Java based monitoring API has been implemented in CICS TG V7.1. It provides a request level monitoring capability. Several samples are included that demonstrate how this API can be used.

A summary of the function provided by the API

- ▶ Exits can be stacked, allowing for multiple request monitors on the same JVM.
- ▶ Method calls are driven when each transaction enters and exits the Gateway daemon (or entry and exit to the CICS TG classes when running in local mode or in a remote application). They are issued at the points E1 and E2 in Figure 6-3.

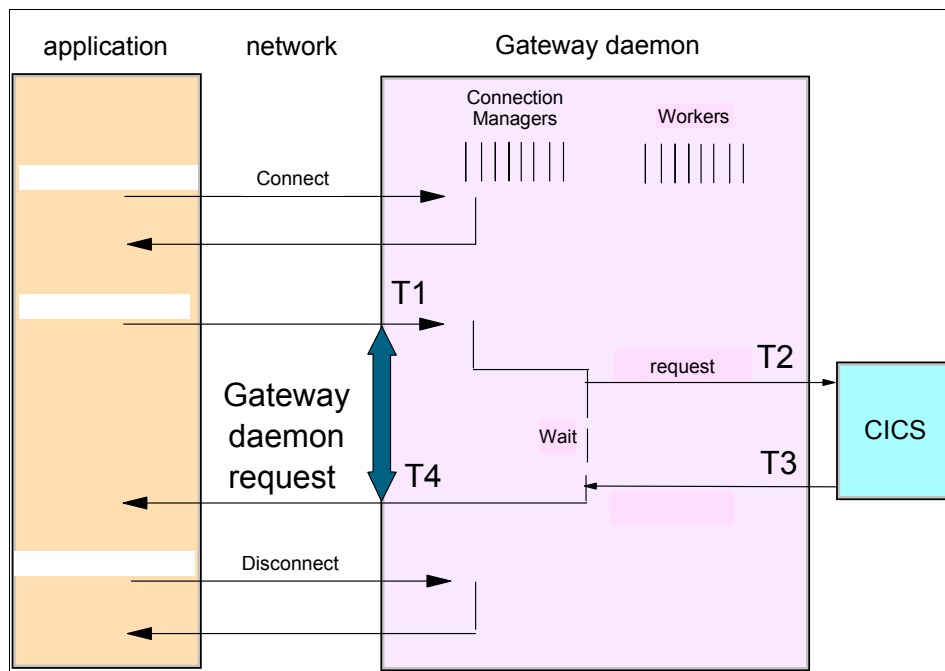


Figure 6-3 The points where time stamps are taken by the exits

- ▶ The API can be driven from both an application address space and the Gateway daemon address space. Both remote mode and local mode applications can be monitored.
- ▶ Time stamps are taken at key boundary points within the application or Gateway daemon processing, making it possible to measure the times spent in the major components of a system.

- Correlation points are available in the exits that uniquely identify each transaction and make it possible to tie together entry and exit flows.

Attention: For XA transactions, the XID is available for use as a correlator. For applications running under WebSphere with EWLM enabled, the EWLM token can be used. For other EXCI sync-on-return applications, a CICS TG specific correlator is generated, which is of a similar format to a CICS LU6.2 UOWID

- Details of each request are made available. The data available includes:
 - Transaction type
 - Request type
 - COMMAREA data
 - Client application IP address
 - Gateway daemon URL
 - CICS server name
 - User ID
 - Tranname and TPN
 - Topology of flow (whether its a local or remote application, Gateway daemon, and so on)
 - Return codes
 - The amount of data sent over the network
 - Time spent waiting for a Worker thread to be allocated to run the request

For full details of all the information available in these exits and the architecture of the API, see the CICS TG V7.1 Information Center.

An overview of the function provided by the samples

We used the ThreadedMonitor and StdoutMonitor sample exits. They output details on each transaction that goes through the address space running the monitoring exits. They also calculate the time taken by the various components of a request.

ThreadedMonitor off loads writing the results to a background thread. StdoutMonitor writes to the output device in line in the exit code. Therefore, the performance impact on the Gateway daemon of running ThreadedMonitor is less than StdoutMonitor. ThreadedMonitor stores the results of the initial request exit and waits until the response exit is driven for that request before outputting the data. This makes the output easier to read because the output from each request is sequential. (StdoutMonitor interleaves requests.)

The time taken for various stages in the processing of each transaction are output in the offset fields. Offset is the time in milliseconds relative to when the request entered the Gateway daemon. The full set of timings is output in the ResponseSent entry point (see Example 6-4).

Example 6-4 ThreadedMonitor output

```
Data[000000B2]: RequestReceived (1191932548688) = Tue Oct 09 08:22:28
EDT 2007 offset = 0 (A)
Data[000000B2]: RequestSent (1191932548688) = Tue Oct 09 08:22:28 EDT
2007 offset = 0 (B)
Data[000000B2]: ResponseReceived (1191932548889) = Tue Oct 09 08:22:28
EDT 2007 offset = 201 (C)
Data[000000B2]: ResponseSent (1191932548889) = Tue Oct 09 08:22:28 EDT
2007 offset = 201 (D)
```

Table 6-2 correlates the offsets recorded by the monitoring exits in Example 6-4 to the time stamps in Figure 6-3 on page 202.

Table 6-2 Correlation of offsets in output with time stamps

Offset (Example 6-4)	Time stamps (Figure 6-3 on page 202)
RequestReceived (A)	T1
RequestSent (B)	T2-T1
ResponseReceived (C)	T3-T1
ResponseSent (D)	T4-T1

Tip: We calculated the time taken for the request from the data in Example 6-4 on page 206. See Table 6-2 to see what A, B, C, D, T1, T2, T3, and T4 refer to.

The total transaction time between entering and leaving the Gateway daemon is $T4 - T1 = D = 201$ ms.

The time taken for the request in CICS is $T3 - T2 = (T3 - T1) - (T2 - T1) = C - B$.

$(C) - (B) = (201 - 0) = 201$ ms

The time taken within the Gateway daemon is the difference between the time between entering and leaving the Gateway and the time spent in CICS.

Tip: To quickly find transactions that are taking a long time to complete in the file, use a filter on the lines containing `offset=`. For example, on UNIX, you could use **grep**. Scan the resultant output for a large offset number. Cross reference the time stamp in the output to the original file to discover details about the transaction that is taking a large amount of time.

Summary of key sample configuration parameters

The only option available to StdoutMonitor is where the output data is sent. There are a number of options for ThreadedMonitor for tuning the performance of the sample and logging when transactions exceed timeout limits.

The options are configurable through JVM system properties. Some of the properties that are available are:

- ▶ `com.ibm.ctg.samples.requestexit.out`
Configures the data output destination to an HFS or zFS file. It defaults to the standard output stream.
- ▶ `com.ibm.ctg.samples.requestexit.err`
Configures the error log to go to an HFS or zFS file. It defaults to the standard error stream.

Tip: When the Gateway daemon is invoked through CTGBATCH, the standard output and error streams can be directed to JES or to a dataset. If the samples are not configured to send the data to zFS files, their output will contain the information and warning messages that the Gateway daemon writes to the standard output and error streams.

- ▶ `com.ibm.ctg.samples.requestexit.lrt`
This is the long running transaction timeout. If a transaction response time exceeds this value, an alert message will be written to the error destination.
- ▶ `com.ibm.ctg.samples.requestexit.reaper` and `com.ibm.ctg.samples.requestexit.orphan`
Request data and response data is interleaved for an individual transaction. These parameters make it possible to configure the amount of time ThreadedMonitor will store the request data for a hung transaction.

See the samples source code for the other configuration parameters.

The impact of using the monitoring samples

We ran a few quick tests to estimate the impact of running the samples. We ran a high workload that the CICS TG could handle comfortably and looked at the differences in response time and CPU with and without the exits.

We configured the Gateway daemon to have 100 Worker manager and Connection Manager threads and drove 100 concurrent connections. Each CICS transaction took 500 ms to complete on CICS. Each connection issued back to back transactions with a think time between each transaction of 100 ms. This drove a transaction rate of around 160 transactions per second.

We measured the CPU used by the Gateway daemon using OMEGAMON XE. To calculate the average, we waited until the workload was constant, took five samples of CPU usage, and calculated the average for these five values. The results are summarized in Table 6-3.

Table 6-3 Monitoring samples impact

Monitoring exit status	Average CPU (%)	User response time (ms)
Off	1.8	0.501
StdoutMonitor	5.1	0.506
ThreadedMonitor	4.9	0.501

We saw that the threaded monitor exits made only a small difference to response time. ThreadedMonitor gave the best response time.

The CPU increase was more significant. We saw that CPU more than doubled, although the CPU usage was still manageable. ThreadedMonitor had a lower CPU impact. Other testing has shown that most of the CPU usage from running the samples code is in writing the data to disk, rather than in running the monitoring exits themselves.

It is worth noting that the impact of running the monitoring exits is much less than running a full product trace. See 6.3.3, “Gateway daemon file system I/O constraint” on page 222 for an idea of how trace affected the Gateway daemon performance.

Attention: The sample applications shipped with the CICS TG are designed to illustrate the functionality available with the monitoring API. They are not designed to be used on production level systems.

Tip: Use ThreadedMonitor when you wish to reduce the performance impact of monitoring or if you want to be alerted if transactions exceed a time limit. Use StdoutMonitor if you need to investigate a transaction hang.

Our test case ran for approximately five minutes and produced approximately 100 MB of output from the monitoring exit. So we needed to budget for approximately 2 MB per minute of zFS space when running under load. The amount of data output depends on the parameters associated with the transactions.

Attention: Make sure that there is sufficient space on the HFS before activating the monitoring exits.

Tip: If you are interested in the alerts rather than the output, a quick way to discard the output data is to send it to the file /dev/null.

The source code for the samples is provided with the CICS TG. It is possible to modify the samples for your own use if you would prefer that it outputs less data. For example, the samples could be modified to only output transaction data if it exceeds the long running transaction timeout.

Configuring the Gateway daemon to invoke the samples

Do the following steps:

1. We set the CLASSPATH used by the CICS TG to include ctgsamples.jar.

A precompiled version of the samples are included in ctgsamples.jar in <install_dir>/classes. To give the Gateway daemon access to this jar file, we added the following to our stdenv:

```
CLASSPATH=/usr/lpp/cicstg/ctg710/classes/ctgsamples.jar
```

2. We specified that the relevant exits were to be loaded when the Gateway daemon started.

Exits are specified through the requestexits field in the GATEWAY section of the Gateway daemon ini file. To use ThreadedMonitor, we set requestexits=com.ibm.ctg.samples.requestexit.ThreadedMonitor, and to use StdoutMonitor we set requestexits=com.ibm.ctg.samples.requestexit.StdoutMonitor.

Tip: Multiple exits can be defined by separating them with a comma. See the Configurations for a user exit implementation topic in the CICS TG Information Center for details on stacking user exits.

3. We configured the sample properties.

We specified that the samples send their output to a zFS file (instead of defaulting to stdout). For ThreadedMonitor, we allowed alert messages to go to their default destination of stderr and set it to write a message if a transaction response time exceeded 1 second.

The samples are configured by JVM system properties. To set a system property through the CICS TG, specify -j-D on the command line. Command line options can be specified by CTGSTART_OPTS in the STDENV file. The values we specified were:

```
CTGSTART_OPTS=-j-Dcom.ibm.ctg.samples.requestexit.out=/ctg/scsctg71/
scstg712/logs/ThreadedMonitor.txt
-j-Dcom.ibm.ctg.samples.requestexit.lrt=1000
```

4. We restarted the CICS TG Gateway daemon.

The message output to the Gateway daemon stdout log indicating that the exit had been loaded is shown in Example 6-5.

Example 6-5 A monitoring exit successfully started

```
[0] CTG8402I Request Monitoring Exit
com.ibm.ctg.samples.requestexit.ThreadedMonitor is enabled
```

6.3 Problem scenarios

We detail the problem setup, the warning indications that would indicate to an operator or systems administrator that a problem occurred, the diagnosis steps, and the actions we took as a result. We document the following problem scenarios:

1. Network delay
2. Gateway daemon thread constraint
3. Gateway daemon I/O constraint
4. CICS constraint

6.3.1 Network delay

This scenario details how we simulated and diagnosed a network constraint issued on the connection between the Java client and the Gateway daemon.

Setup

Table 6-4 shows the setup information for the Network delay.

Table 6-4 Setup information for Network delay

Key parameter	Setting
Maximum Connection Manager threads per Gateway daemon	100
Maximum Worker threads per Gateway daemon	100
Approximate CICS transaction time	0 ms
Number of concurrent connections from the workload driver	100
Think time between requests on each connection on the workload driver	0.1s

We introduced network delays by driving the workload from the front-end application on a computer that was geographically dispersed from the LPAR running the CICS TG and CICS systems (see Figure 6-4). In our test systems, this increased average transaction response times from several milliseconds to several hundred milliseconds.

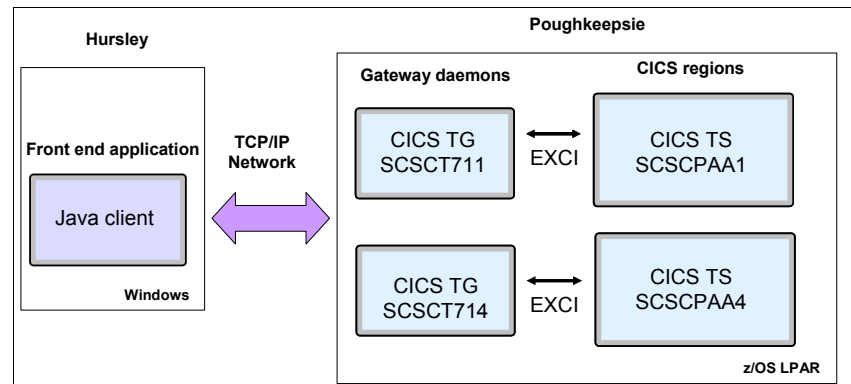


Figure 6-4 Simplified setup for a high network latency

Warning indications

The user reports a consistently slow response time. The Gateway daemon average response time GD_IJAVRESP is faster than the response time reported by the user. There are no OMEGAMON XE alerts indicating resource shortages within the CICS TG or CICS or z/OS.

Problem diagnosis

We ran **ctgping** on the computer that was driving the workload. First of all, we set it to ping the Gateway daemon only; we then set it to run transactions to CICS through the Gateway daemon. Response times were compared. Table 6-5 summarizes the results.

Table 6-5 Summary of ctgping results

Ping request	Average Open (ms)	Average Request (ms)	Average Close (ms)
Gateway daemon	200	105	93
CICS region	200	103	97

The open and close time shown in Table 6-5 are relatively slow. In normal circumstances on a high speed network most ping flows should take less than 10 ms; on our network, running **ctgping** in the same LPAR as the CICS TG gave an average open time of 4 ms, an average request time of 2 ms, and an average close time of 1 ms. We can also see that the average request time does not increase when the ping request is sent to CICS. In fact, sending a ping request to a CICS region is actually quicker than the list systems processing, which involves the Gateway daemon ping!

Tip: A rough estimate of average time spent in CICS is as follows:

Average response running CICS transaction - Average response for list systems

This formula assumes that the list systems processing time is negligible compared to the CICS transaction time and that the code path of issuing a list systems within the Gateway daemon is similar to running a transaction. We found list systems requests to take a few milliseconds, so this formula is only useful if the CICS transaction time is expected to be larger.

Because the results indicate that the CICS system is responding to requests very quickly, we need to look outside of CICS for the source of the slow down. To see whether the Gateway daemon component could be a source of the slow request time, we looked at some key CICS TG statistics shown in Table 6-6 on page 213.

Table 6-6 CICS TG response time statistics

Stat name	Value (ms)
GD_IAVRESP	3
CS<SERVERNAME>_IAVRESP	0

The Gateway daemon component recorded its average request time as 3 ms. This is 102 ms less than the **ping** request time.

Tip: The average time spent in Gateway daemon, excluding time spent in CICS, is equal to GD_IAVRESP - CS_IAVRESP without any extra qualification.

Because the Gateway daemon response is very quick, we need to consider the network component as the source of the delay. The TCP/IP **ping** tool that comes as part of the operating system was invoked from the Hursley machine to the target system in Poughkeepsie. It took 98 ms to reply.

Action taken

All evidence points to network latency causing the delay. The front end was moved to a system in Poughkeepsie with faster network access. Transaction request response times improved from 106 ms to 6 ms as a result.

6.3.2 Gateway daemon Worker thread constraint

Here we discuss the Gateway daemon Worker thread constraint.

Setup

Table 6-7 shows the key setup configuration parameters and base workload.

Table 6-7 Key setup configuration parameters and base workload

ini file setting	Key parameter	Setting
maxconnect	Maximum Connection Manager threads per Gateway daemon	500
maxworker	Maximum Worker threads per Gateway daemon	100

ini file setting	Key parameter	Setting
workertimeout	Workertimeout	1000 ms
	Approximate CICS transaction time	500 ms
	Number of concurrent connections from the workload driver	500
	Think time between requests on each connection on the workload driver	1s

Attention: We set workertimeout to 1000 ms. This means that the Gateway daemon will timeout a request if it has to wait more than 1 second for a Worker thread to become available.

OMEGAMON XE for CICS TG on z/OS comes with a number of sample CICS TG alerts. Because users generally want to tailor the alerts based on the qualities of service they expect from the system, the majority of these alerts are switched off by default. We enabled them for the duration of this test.

Tip: The only CICS TG alert that is active by default is the alert that is driven when the CICS TG health reaches 0.

We set the system up as above and ran the base workload continuously. In order to produce a Worker thread constraint, we then started another workload on top of the existing one (Table 6-8).

Table 6-8 First additional workload

Key parameter	Setting
Number of concurrent connections from the workload driver	300
Think time between requests on each connection on the workload driver	1s

To push the constraint further, we added a further workload in addition to the other two (Table 6-9 on page 215).

Table 6-9 Second additional workload

Key parameter	Setting
Number of concurrent connections from the workload driver	100
Think time between requests on each connection on the workload driver	0.1s

We kept the CICS transaction time constant at 500 ms for each of the workloads.

Warning signs

The user reports an increase in transaction response time. The Gateway daemon stat CM_CWAITING shows that Connection Managers are waiting for Worker threads. An OMEGAMON CICSTG_ConnWait_Warning may be driven at this point if it is active.

As workload increases further, and waits for Worker threads to become larger than workertimeout specified in ctg.ini, users start to complain about requests failing. WT_ITIMEOUTS starts to increase. An OMEGAMON CICSTG_WorkerTimeout_Warning may be driven at this point if it is active.

Diagnosis

The workload changed during the course of this scenario. The duration of changes in workload was around 20 minutes. This is less than the interval period of 1 hour. So if the response times slow down during the time period, the averages will not show the full delay. They will show the average since the last interval, rather than the average over the last 10-20 minutes when workload has increased.

Even if we do not use the absolute values, the averages can still be useful. We noticed that the average response time for the Gateway daemon GD_IAVRESP increased as the workload increased, while the average CICS response time stayed more or less constant. This tends to indicate that the CICS TG is not dealing with the increased workload as well as the CICS server. So we decided to investigate the CICS TG component first.

Table 6-10 through Table 6-12 show the CICS TG resource usage at various stages in the workload.

Table 6-10 Resources with base workload

Gateway daemon	CM_CALLOC	CM_CWAITING	WT_CALLOC	WT_LTIMEOUTS
SCCSCT711	252	0	91	0
SCSCT714	248	0	81	0

Table 6-11 Resources with base workload plus first set of workload

Gateway daemon	CM_CALLOC	CM_CWAITING	WT_CALLOC	WT_LTIMEOUTS
SCCSCT711	352	59	100	2
SCSCT714	348	39	99	3

Table 6-12 Resources with all three workloads running simultaneously

Gateway daemon	CM_CALLOC	CM_CWAITING	WT_CALLOC	WT_LTIMEOUTS
SCCSCT711	400	140	100	341
SCSCT714	400	118	100	310

The OMEGAMON XE portal showed the workload in a more intuitive way than the raw statistics. Figure 6-5 shows the Connection Manager usage where we have the base workload plus one extra set of workload. We can see straight away that Connection Managers are waiting for Worker threads to become free.

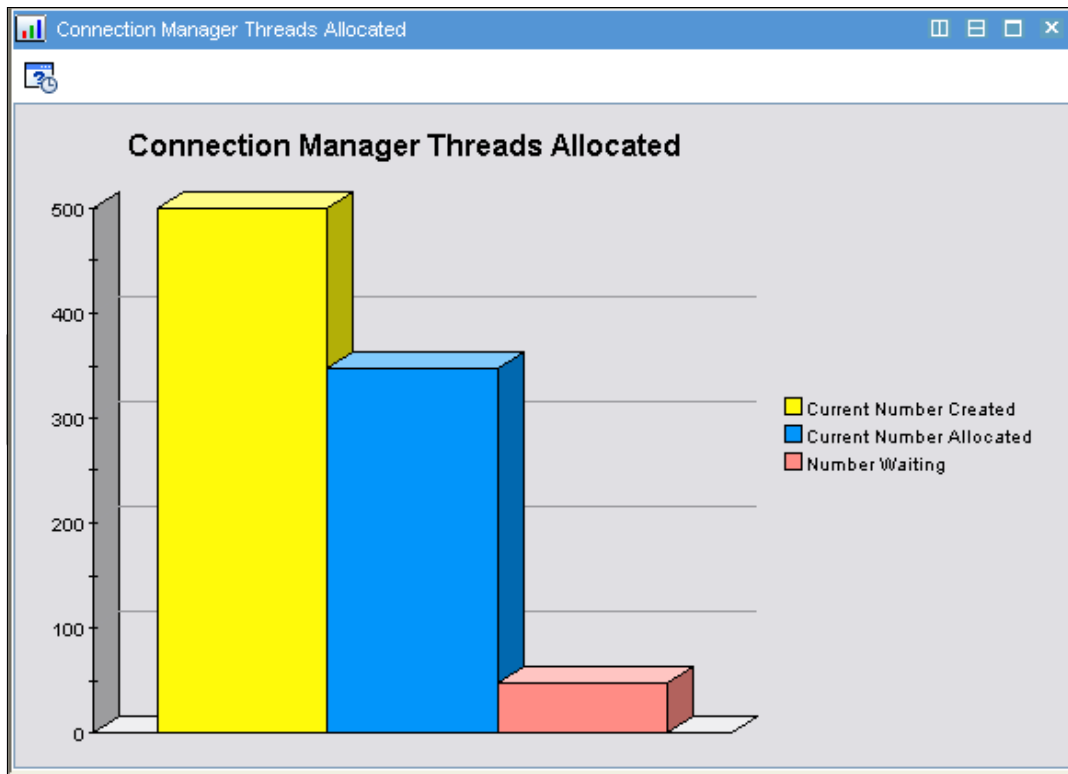


Figure 6-5 Base workload and first extra workload

When we look at the Worker thread usage (Figure 6-6), this immediately confirms that we are running at full Worker capacity.

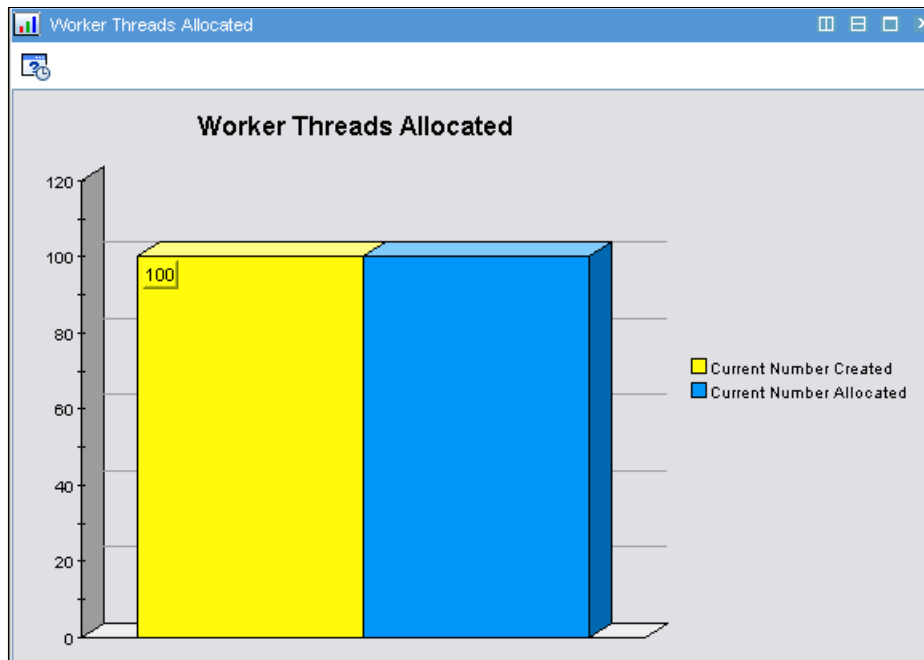


Figure 6-6 Running base workload + additional workload: Workers at full capacity

When the additional workloads were added, we saw a number of OMEGAMON alerts. There were a number of CTG_FreePipeWarning and CTG_WorkerAllocWarning alerts prior to the timeouts occurring and requests failing (Figure 6-7).

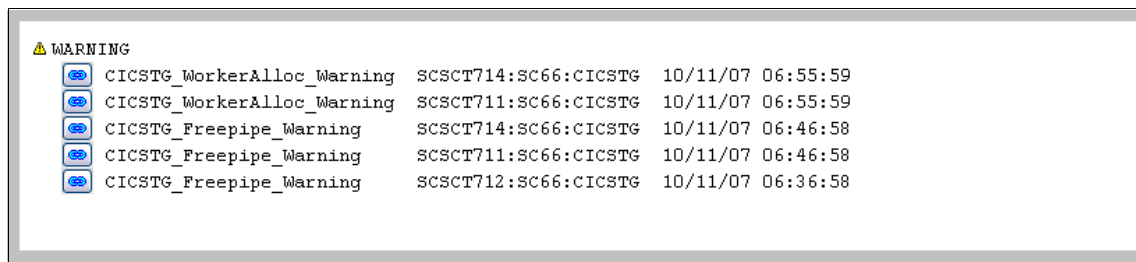


Figure 6-7 Warnings produced under high workload

We configured the Gateway daemons so that they had CTG_PIPE_REUSE=ONE and had a maximum of 100 workers. Therefore, our Gateway daemon configuration cannot run out of EXCI pipes.

(CTG_PIPE_REUSE=ONE specifies a maximum of one pipe per Worker.) The CICSTG_FreePipe_Warning indicates that we have less than 10 free pipes. It means that the CICS TG is running at a high load but it does mean that a pipe allocation failure will occur with our configuration. The pipe usage is a high watermark. Once the Gateway daemon has allocated a pipes, it caches it. Therefore, once we get into this state, we will continually be in a “no free pipes state”.

CICSTG_WorkerAllocWarning indicates that 90% of the maximum workers are allocated. This is an indication that the CICS TG is at high load, and if the number of concurrent requests were to increase by more than 10%, we would likely start waiting for Worker threads to become free.

Action taken

We re-configured the system so that the problem no longer occurred and we modified the alerts to give us more indication of the warnings. An example of changing the system setup to deal with extra capacity is covered in 8.3, “Using historical data to diagnose system problems” on page 297.

The number of Worker threads that we expect to be in use depends on the dynamics of the particular system setup. When CICS transactions complete very quickly we would not see an impact if requests were waiting for Worker threads to become free. For example, if the transaction took 50 ms in CICS, each Worker would be freed in a little over 50 ms, so several Connection Manager threads could queue up on a Worker without a noticeable delay. In this case, we might even want to reduce the number of Worker threads to reduce EXCI pipe usage and to reduce the storage requirements of the Gateway daemon.

In our case, we have a CICS transaction time of 0.5 second. If we want to have a sub second response time, we cannot afford to have requests back up waiting for Worker threads. Our timeout is driven when we have to wait more than 1 second. We decided to configure an alert to give us advance warning before a timeout is driven (and users start to notice errors occurring).

Tip: The performance section of the CICS Transaction Gateway Information Center provides a rule of thumb to help estimate the number of Worker threads needed. If you are aiming to support a specific transactions per second rate (TPS) through the Gateway daemon, then:

numW = number of Worker threads, *TPS* = transactions per second, and *CICSRespTime* = average response time through CICS

$$numW \geq TPS \times CICSRespTime$$

The average CICS response time can be measured from CS_LAVRESP (remember to convert it from ms to seconds).

If we do not know the TPS, we can estimate it by using this formula:

numUsers = number of concurrent users, *thinkTm* = average user think time, and *GDRespTime* = Gateway daemon response time.

$$TPS = numUsers / (thinkTm + GDRespTime)$$

If we substitute that formula into the previous formula, we get:

$$numW \geq GDRespTime * numUsers / (thinkTm + CICSRespTime)$$

The average Gateway daemon response time can be measured using GD_LAVRESP (remember to convert it from ms to seconds).

The above formulas are intended as a very rough general rule. Because all systems have different dynamics, this should be used as a starting point for tuning the system only. It does not guarantee that the threads are correct for the quality of service required. All timings are in seconds in the equations.

Consideration: The following assumptions were made when deriving the formulas above:

- ▶ When an optimal number of workers is configured, the time taken for the request running through the Gateway is small.
- ▶ The CICS response time stays constant as workload varies.

We configured the sample CICSTG_ConnWait_Warning to be driven when we were approaching the threshold where Connection Managers time out. By default, CICSTG_ConnWait_Warning is driven when the Connection Managers waiting for a worker is 90% of the total Connection Managers allocated.

We used some very rough thumbnail calculations to estimate a new threshold for this alert, based on our results.

In Table 6-8 on page 214, the requests waiting per total connections was 16%.

Tip: $CM_WAITING/CM_CALLOC = 59/352 = 17\%$

In Table 6-9 on page 215, the requests waiting per total connections was 0.35%.

Tip: $CM_WAITING/CM_CALLOC = 140/400 = 35\%$

There were only a few worker timeouts when we drove the first additional workload (Table 6-8 on page 214). When the second workload was applied, we saw a large number of timeouts (Table 6-9 on page 215). We decided to set the CTG_ConnWait_Warning somewhere between the two values to hopefully give us some warning in advance of large numbers of requests timing out.

We re-configured the alert to 20% and reran the workload. We saw that we got the ConnWait warning before the WorkerTimeout warning (see Figure 6-8).

By clicking the button to the left of the alert text, we could see details about the situation values that drove the alert (see Figure 6-9 on page 222).

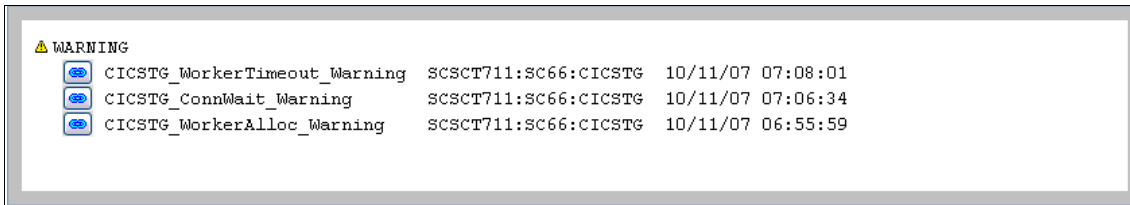


Figure 6-8 ConnWait timeout alert at 20%

Percentage Waiting	Origin Node	System ID	Gateway Daemon Name	Thread Limit	Current Number Created	Current Number Allocated	Tc
29	SCSCT711:SC66:CICS...	SC66	SCSCT711	500	500	400	

Percentage Waiting	Origin Node	System ID	Gateway Daemon Name	Thread Limit	Current Number Created	Current Number Allocated	Tc
25	SCSCT711:SC66:CICSTG	SC66	SCSCT711	500	500	400	

Figure 6-9 Situation values for the ConnWaiting alert we configured

6.3.3 Gateway daemon file system I/O constraint

Here we discuss the Gateway daemon file system I/O constraint.

Setup

Table 6-13 shows the key parameters for the HFS I/O constraint.

Table 6-13 Key parameters for the HFS I/O constraint

Key parameter	Setting
Maximum Connection Manager threads per Gateway daemon	100
Maximum Worker threads per Gateway daemon	100
Approximate CICS transaction time	0 ms
Number of concurrent connections from the workload driver	100
Think time between requests on each connection on the workload driver	0.1s

While the workload was running, JNI trace and full debug Gateway daemon trace was switched on for the Gateway daemon.

Warning signs

The Tivoli Enterprise Portal shows that I/O Per minute in CICS TG summary for the Gateway daemon process increases from a two digit number to a five digit number. CPU percentage increases at the same time Requests per minute decreases.

Diagnosis

We monitored the average response time statistics to look for any delays in the Gateway daemon. The response time of the Gateway daemon increased from 1 ms to 16 ms, as shown in Table 6-14.

Table 6-14 Response time versus trace status

Trace status	GD_IAVRESP (ms)	CS_IAVRESP (ms)
Off	1	0
On	16	0

We also monitored the following fields in the Gateway daemon overview section of the TEP workspace:

- ▶ I/O per minute
- ▶ Requests per minute
- ▶ CPU utilization

Our findings are shown in Table 6-15.

Table 6-15 OMEGAMMON XE stats versus trace status

Trace status	I/O per minute	CPU utilization	Request per minute
Off	0	9%	49,150
Full debug Gateway daemon trace plus JNI trace	241,000	65%	4000

A sharp decrease in requests per minute at the same time as a CPU increase indicate that transaction processing is now being constrained within the CICS TG. The I/O increase gives us a clue as to why the delay is occurring. It indicates a large amount of logging is taking place in the Gateway daemon address space.

Tip: OMEGAMON XE calculates the I/O per minute statistic by taking the current number of I/Os in the CICS TG over the most recent sampling interval, divided by the duration of the interval (from 60 to 120 seconds), to give the average over the past minute.

We configured the CICS TG to write logs to JES and traces to HFS. We viewed the JES logs associated with the CICS TG job under SDSF. No new records were being written to the JES log at the time the problem occurred. This left tracing as a likely culprit for the high I/O.

We issued the z/OS modify command shown in Example 6-6 against the Gateway daemon to verify whether tracing was active or not.

Example 6-6 Trace status enquiry

```
/F SCST712,APPL=TRACE
```

```
RESPONSE=SC66
BPXM023I (CTGUSER)
CTG8239I Response received from CICS Transaction Gateway
Gateway Daemon trace settings:
    tlevel=4
    truncationsize=80
    dumpoffset=0
    tfile=/ctg/scsctg71/logs/ctg.trc
    tfilesize=0
JNI trace settings:
    jnilevel=1
    jnifile=/ctg/scsctg71/scstg712/logs/jni.trace
```

The response of tlevel=4 and jnilevel=1 indicate that full tracing is active. When tracing is not enabled, values for 0 are expected.

Action taken

We deactivated tracing. Performance returned to the accepted level.

To give us advance warning of any possible problems in future, we decided to create an alert on OMEGAMON XE to indicate whether a Gateway daemon was doing an excessive amount of logging.

First, we estimated the I/O impact per JES message. This is two I/O counts per message.

Tip: To get a rough estimate of the I/O count increase due to an individual log message, do the following steps:

1. Refresh the OMEGAMON XE workspace with PF5 and note the current I/O count on an idle CICS TG.
2. Issue a **ctgping** command to drive one connect and disconnect from the Gateway daemon, for example:

```
ctgping -i=1 -r=0 wtsc66.itso.ibm.com:2007
```

3. Verify that two messages have been logged as a result of this in the JES STDOUT log:

```
10/02/07 13:49:22:812 [0] CTG6506I Client connected.
```

```
[ConnectionManager-63]
```

```
10/02/07 13:49:23:015 [0] CTG6507I Client disconnected.
```

```
[ConnectionManager-63]
```

Refresh the OMEGAMON XE workspace again and note the change in I/O count.

In our base setup, the CICS TG is configured to log each time an application connects and disconnects. We only want an alert if the number of messages indicates that we have inadvertently activated trace and are therefore impacting performance. We decided to set the alert level to 10,000. Having it this high makes the alert unlikely to be triggered as a result of a large number of applications suddenly connecting to the Gateway daemon.

We created a situation in the following way using the Tivoli Enterprise Portal GUI:

1. Select the CICS TG instance the alert applies to in the Enterprise explorer pane.
2. Right-click the instance and select **Situations**.

3. Click the **Create Situation** icon in the top left hand corner, as shown in Figure 6-10.

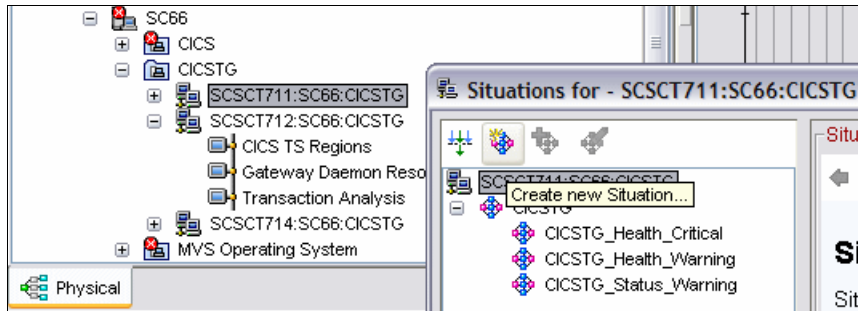


Figure 6-10 Create situation

4. A Create Situation pop-up menu appears, as shown in Figure 6-11.

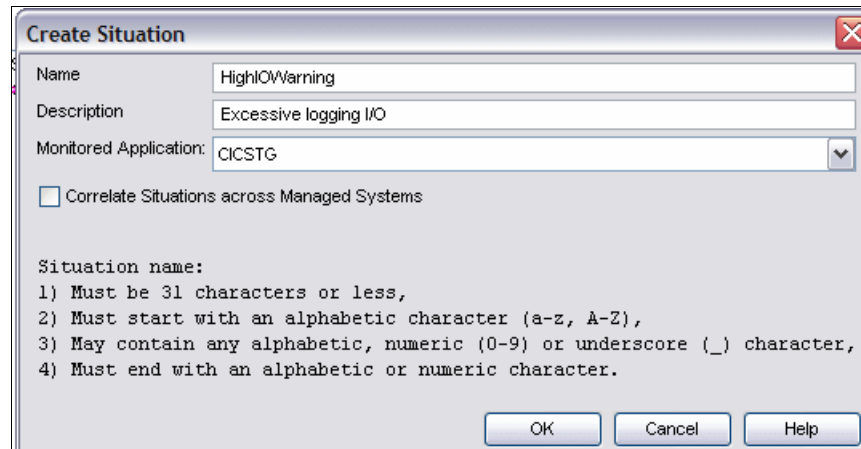


Figure 6-11 HighIOWarning

5. Enter a name and description for the situation. (We chose HighIOWarning.)

Attention: If the OK button is greyed out, check that you have not included restricted characters, for example, spaces, in the name.

6. Click **OK**. Select the attribute that will be used to trigger the alert. We chose I/O per minute, as shown in Figure 6-12.

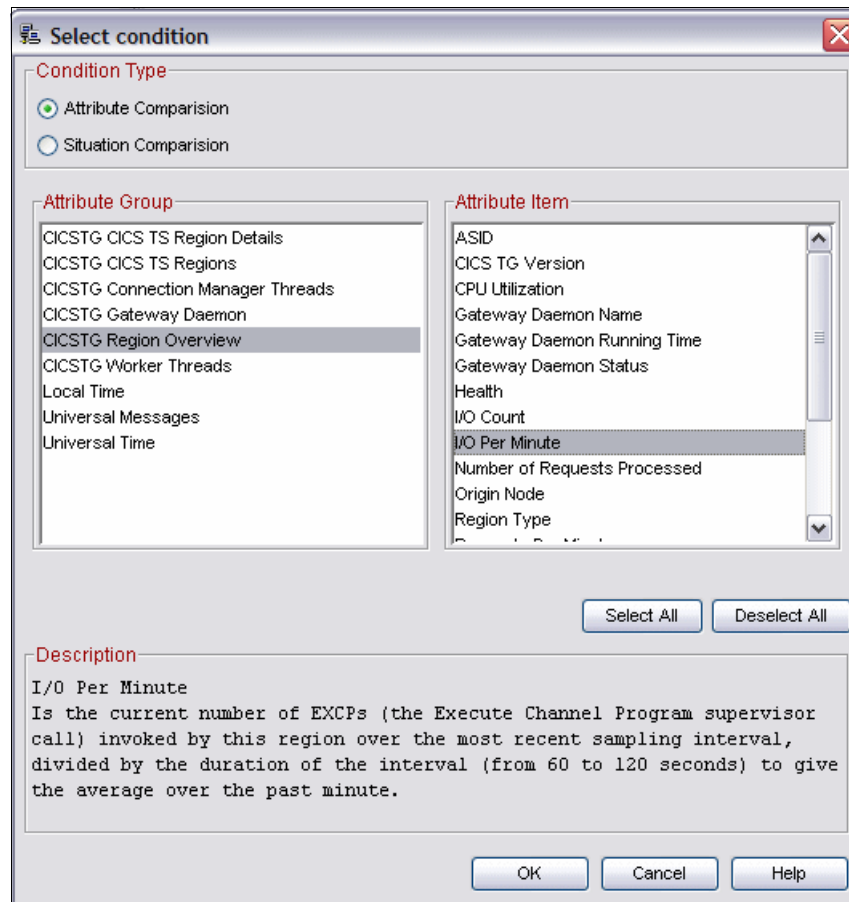


Figure 6-12 Condition selection

7. The box shown in Figure 6-13 appears. This box allows you to create a rule to trigger the alert. Click **1** to create the first rule. Click the equals sign to get a selection of conditions that can be used. We selected greater than and entered 10,000. This meant that the trigger would occur if I/O per minute' was greater than 10,000.

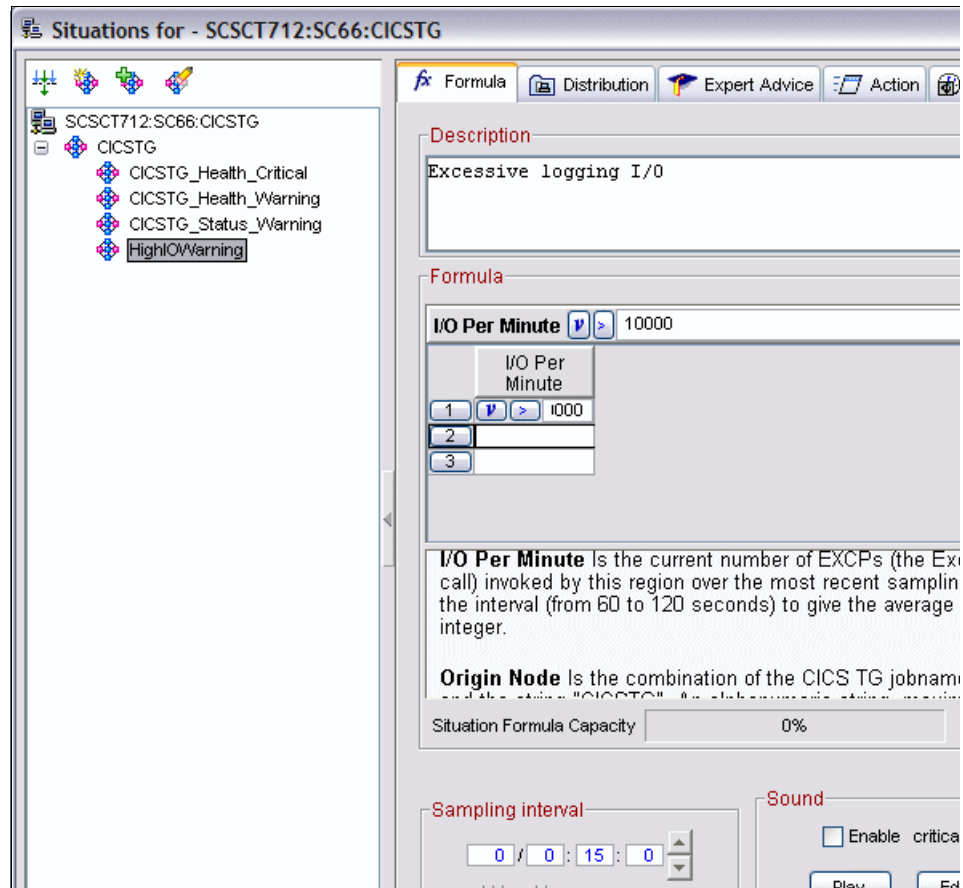


Figure 6-13 Situations formula window

We then re-ran the problem scenario. An alert was produced on the Tivoli Enterprise Portal (TEP) that let us know that a problem had occurred.

Figure 6-14 shows the alert appearing on the Gateway daemon pane. A yellow triangle with an explanation mark in the center indicates that the alert occurred.

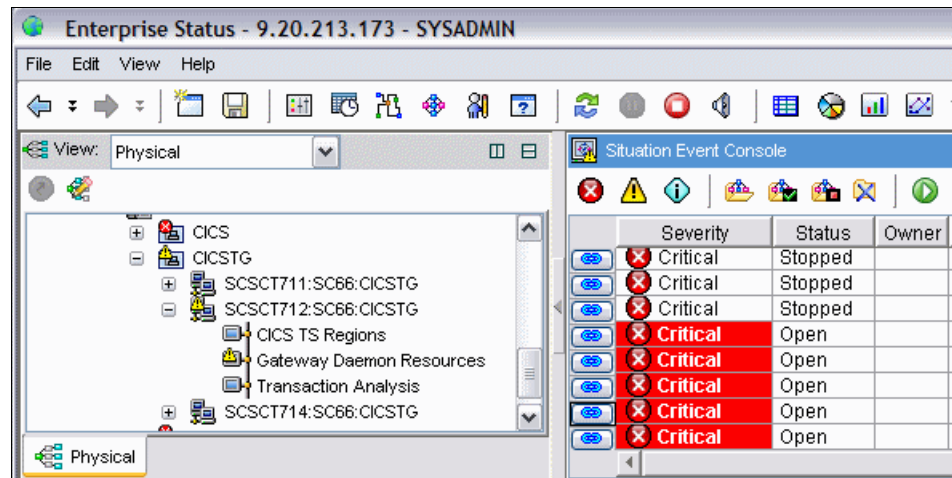


Figure 6-14 An alert in the Gateway Daemon physical view window

Figure 6-15 shows more details of our HighIOWarning alert in the Situation Event Console.

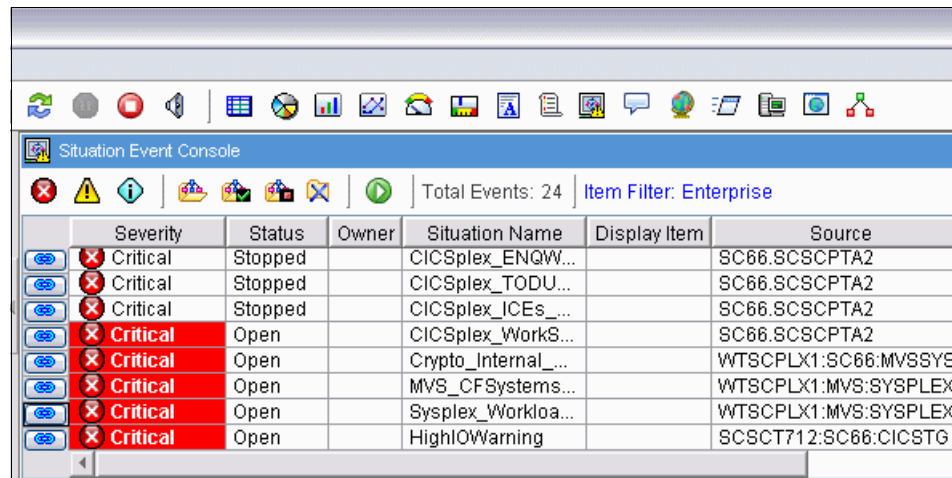


Figure 6-15 Situation Event Console HighIOWarning

6.3.4 CICS constraint

Here we discuss the CICS constraint.

Setup

Table 6-16 shows the key parameters for the CICS constraint scenario.

Table 6-16 Key parameters for the CICS constraint scenario

Key parameter	Value
Maximum Connection Manager threads per Gateway daemon	100
Maximum Worker threads per Gateway daemon	100
Approximate CICS transaction time	100 ms
Number of concurrent connections from the workload driver	100
Think time between requests on each connection on the workload driver	0.1s

Instead of using the CICS transaction that returned immediately, a different transaction was used that had a 100ms delay.

On one of the CICS regions, the correct group was installed with the program definitions. The program definition for the delay transaction was threadsafe. On the other CICS region, we did not install the group. This means that the program was autoinstalled the first time it was used. The autoinstalled version of the program was installed as *quasi-reentrant*. Quasi-reentrant means that the program can only run on one TCB (the QR_TCB). If multiple concurrent requests are issued for this transaction, they will queue up while waiting to use the QR_TCB.

The workload invoked multiple concurrent versions of the program. Each time the quasi-reentrant version of the transaction was invoked, the QR_TCB was tied up for 0.1 seconds and no other programs could use this TCB. This produced a single threaded bottleneck on the delay program, which caused queuing and large delays in the CICS system.

Warning signs

Some users experience transaction delays or transaction timeouts. The Gateway daemon statistics show that there are Worker threads available and the CPU usage is low.

Diagnosis

The output from our workload driver on the front end showed that the average transaction response time was 0.97s. It showed that 82% of transactions ran in under 0.1s, 1% ran between 0.1s and 0.2s, and 27% ran between 0.2s and 1s. Therefore, although the majority of transactions run quickly, a significant number are delayed.

We took a look at the thread usage and request rate on the Gateways. The values shown in Table 6-17 are a snapshot of when we looked at the system under load.

Table 6-17 CPU and requests per minute

Gateway	CPU%	Transaction rate (request per minute)
SCSCT711	1.3%	7761
SCSCT714	5.1%	35539

This shows us that the workload is not evenly distributed between the Gateway daemons. For some reason, SCSCT714 is running at a higher transaction rate. The CPU usage on SCSCT714 is higher, as would be expected if it is running more transactions. We can also see that the CPU has increased a little less than 5x and the requests per minute has increased to around 5x. So, very roughly, CPU seems to be increasing along with the transaction rate. The same information can be seen more intuitively by the OMEGAMON XE GUI (see Figure 6-16 on page 232).

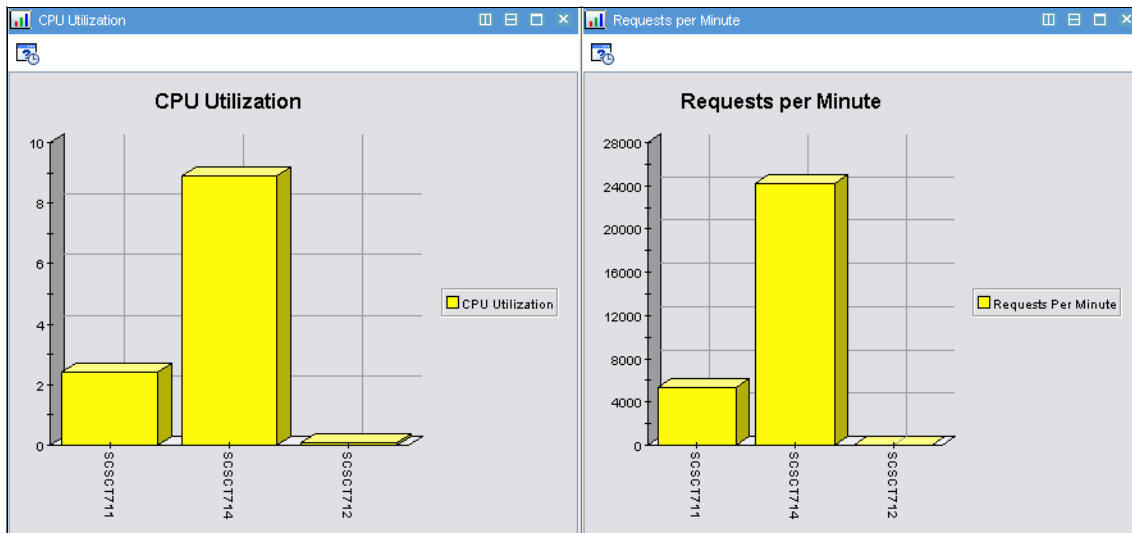


Figure 6-16 CPU Utilization and Requests per Minute graphs

We looked at some further statistics to see if we could understand more about what was causing the uneven distribution of requests, and whether that could be related to our slow transaction times (see Table 6-18).

Table 6-18 Gateway daemon statistics

Gateway	WT_CALLOC	WT_ITIMEOUTS	CM_CALLOC
SCSCT711	44	0	52
SCSCT714	12	0	48

Both Gateway daemons have a nearly equal number of Connection Manager threads (CM_CALLOC). This indicates that the port sharing workload balancing is working correctly (see 6.3, “Problem scenarios” on page 210 for a brief overview on the system setup). Interestingly, the Worker thread allocation is not evenly distributed. SCSC714 (the gateway with the higher CPU and transaction rate) has fewer Worker threads currently in use (WT_CALLOC).

Worker threads are allocated from a pool when the Gateway daemon needs to run a transaction. They remain allocated for the duration of the transaction. When the transaction completes and the response has been sent back to the application, the Worker is returned to the pool. The combination of a low transaction rate and high thread usage tends to indicate that transactions are taking a long time to complete on SCSC711.

We can also see this in a more intuitive format on the OMEGAMMON XE GUI. Figure 6-17 and Figure 6-18 show the thread usage of SCST711 and SCST714, respectively.

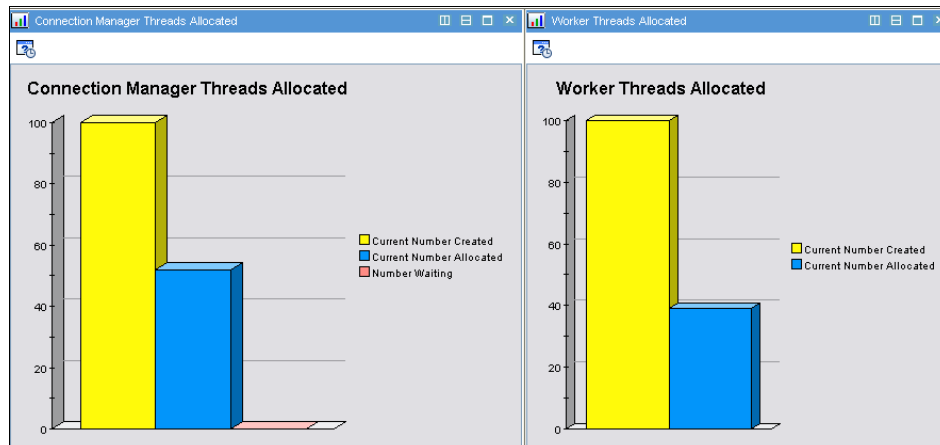


Figure 6-17 SCST711 thread usage

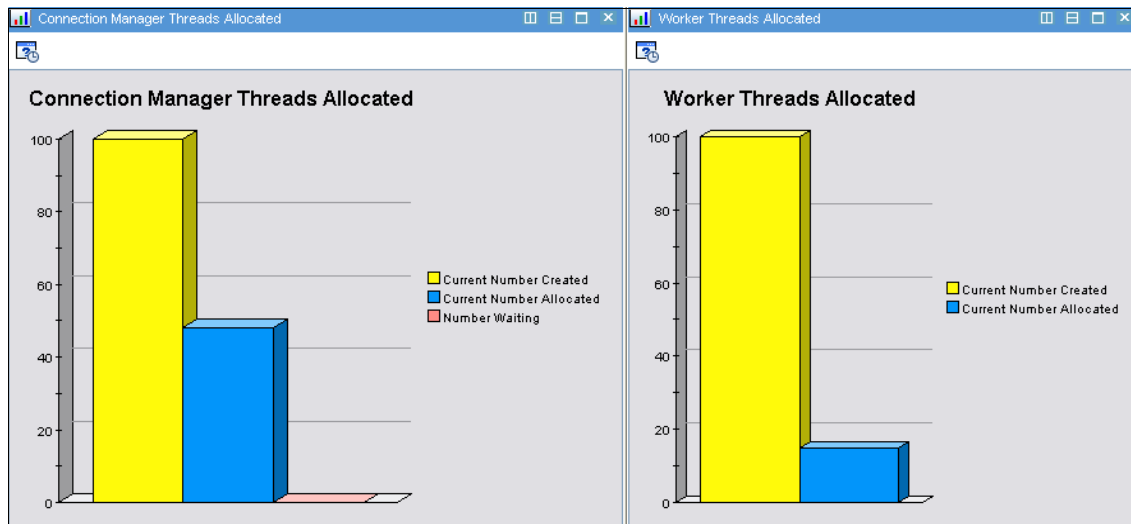


Figure 6-18 SCST714 thread usage

We then looked at the average response times in the CICS TG statistics to confirm our theory about transactions taking a long time to complete on one of the Gateway daemons (see Table 6-19).

Table 6-19 Slow response on one Gateway daemon

Gateway daemon	CS_IHVRESP (ms)	GD_IHVRESP (ms)
SCSCT711	468	468
SCSCT714	11	12

This indicates that the CICS region that SCSCT711 was using has a transaction delay of 468 ms.

The Tivoli Enterprise Portal also provides information about specified CICS regions. We used this to debug where the delay could be occurring in the CICS side. The slow Gateway daemon (SCSCT711) connects to CICS region SCSCPAA1. We selected the Transaction Analysis view for the SCSCPAA1 CICS region. This showed a large number of tasks queued up against the QR TCB (see Figure 6-19 on page 235).

Transaction Analysis											
	System ID	CICS Region Name	CICS SYSIDNT	Transaction ID	User ID	Terminal ID	Task Number	Resource Type	Resource Name	Task State	Elapsed Time
	SC66	SCSCPAA1	PAA1	CSMI	n/a		56865	DISPATCH	QR_TCB	Dispatbl	00:00
	SC66	SCSCPAA1	PAA1	CSMI	n/a		56866	DISPATCH	QR_TCB	Dispatbl	00:00
	SC66	SCSCPAA1	PAA1	CSMI	n/a		56867	DISPATCH	QR_TCB	Dispatbl	00:00
	SC66	SCSCPAA1	PAA1	CSMI	n/a		56868	DISPATCH	QR_TCB	Dispatbl	00:00
	SC66	SCSCPAA1	PAA1	CSMI	n/a		56869	DISPATCH	QR_TCB	Dispatbl	00:00
	SC66	SCSCPAA1	PAA1	CSMI	n/a		56870	DISPATCH	QR_TCB	Dispatbl	00:00
	SC66	SCSCPAA1	PAA1	CSMI	n/a		56871	DISPATCH	QR_TCB	Dispatbl	00:00
	SC66	SCSCPAA1	PAA1	CSMI	n/a		56872	DISPATCH	QR_TCB	Dispatbl	00:00
	SC66	SCSCPAA1	PAA1	CSMI	n/a		56873	DISPATCH	QR_TCB	Dispatbl	00:00
	SC66	SCSCPAA1	PAA1	CSMI	n/a		56874	DISPATCH	QR_TCB	Dispatbl	00:00
	SC66	SCSCPAA1	PAA1	CSMI	n/a		56875	DISPATCH	QR_TCB	Dispatbl	00:00
	SC66	SCSCPAA1	PAA1	CSMI	n/a		56876	DISPATCH	QR_TCB	Dispatbl	00:00
	SC66	SCSCPAA1	PAA1	CSMI	n/a		56877	DISPATCH	QR_TCB	Dispatbl	00:00
	SC66	SCSCPAA1	PAA1	CSMI	n/a		56878	DISPATCH	QR_TCB	Dispatbl	00:00
	SC66	SCSCPAA1	PAA1	CSMI	n/a		56879	DISPATCH	QR_TCB	Dispatbl	00:00
	SC66	SCSCPAA1	PAA1	CSMI	n/a		56880	DISPATCH	QR_TCB	Dispatbl	00:00
	SC66	SCSCPAA1	PAA1	CSMI	n/a		56881	DISPATCH	QR_TCB	Dispatbl	00:00
	SC66	SCSCPAA1	PAA1	CSMI	n/a		56882	DISPATCH	QR_TCB	Dispatbl	00:00
	SC66	SCSCPAA1	PAA1	CSMI	n/a		56883	DISPATCH	QR_TCB	Dispatbl	00:00
	SC66	SCSCPAA1	PAA1	CSMI	n/a		56884	DISPATCH	QR_TCB	Dispatbl	00:00
	SC66	SCSCPAA1	PAA1	CSMI	n/a		56885	DISPATCH	QR_TCB	Dispatbl	00:00
	SC66	SCSCPAA1	PAA1	CSMI	n/a		56886	DISPATCH	QR_TCB	Dispatbl	00:00
	SC66	SCSCPAA1	PAA1	CSMI	n/a		56887	DISPATCH	QR_TCB	Dispatbl	00:00
	SC66	SCSCPAA1	PAA1	CSMI	n/a		56888	DISPATCH	QR_TCB	Dispatbl	00:00

Figure 6-19 Tasks queued against the QR_TCB on SCSCPAA1

CSMI is the mirror transaction. It is the trans ID that CICS TG on z/OS requests run under by default. The QR_TCB is the TCB that CICS transactions run under when they are not thread safe. A TCB is analogous to the thread concept on UNIX System Services. The fact that the transactions are queuing up against the QR_TCB means that they are single threading through CICS.

We issued a CEMT INQUIRE against the program that our workload driver issues. It showed that the program was defined as quasi-reentrant. Figure 6-20 represents the results of the inquire. QUA indicates that the transaction is quasi-reentrant.

```
I PROG(CICDELAY)
,STATUS: ,RESULTS - OVERTYPE TO MODIFY
,Prog(CICDELAY),Leng(0000000472),Ass,Pro,Ena,Pri, ,Ced, ,
, ,Res(000),Use(0000000847),Bel,Uex,Ful,Qua,Cic, ,Nat,

,SYSID=PTA2,APPLID=SCSCPAA1
,RESPONSE:,NORMAL,TIME: 05.41.04 DATE: 10.10.07
PF,1,HELP, ,3,END, ,5,VAR, ,7,SBH,8,SFH,9,MSG,10,SB,11,SF,
```

Figure 6-20 CEMT I PROG against a quasi-reentrant program

Further investigation showed that the program was defined as thread-safe but the resource group containing the program definition had not been installed. This meant that CICDELAY was autoinstalled on the first invocation. The autoinstaller installs it as quasi-reentrant.

Action taken

We installed the resource group with the correct program definition. The CICS region started performing with the expected response times.

We considered the following actions to make similar problems quicker to diagnose in the future:

- Use a range of mirror transaction names instead of relying on the default.

As our test application was only running one program, it was easy for us to tell which program was at fault. In a system with multiple different programs, it might be advisable to have a range of mirror transactions being used. This means that if a transaction is delayed in CICS it is easier to work out which program is responsible.

6.3.5 The delay only affects a minority of the requests

Here we discuss what to do if the delay only affects a minority of the requests.

Setup

Table 6-20 shows the key parameters where the delay only affects a minority of requests.

Table 6-20 Key parameters where the delay only affects a minority of requests

Key parameter	Setting
Maximum Connection Manager threads	100
Maximum Worker threads	100
Approximate CICS transaction time	200 ms and 10,000 ms
Number of concurrent connections from the workload driver	10
Think time between requests on each connection on the workload driver	1s

The setup for this scenario differed significantly from previous scenarios.

We used a cut down setup with only one Gateway daemon. We had 100 Connection Manager and Worker threads defined in the Gateway daemon. We set the back-end CICS program to delay for 200 ms. The workload driver was configured to issue 10 concurrent requests with a think time of 1s between each request.

We introduced another back-end CICS program with a delay of 10s. This was invoked twice during the test run. The EciB2 sample, which ships with the CICS TG install, was used to invoke the slower back end.

The CICS TG was configured with ThreadedMonitor active. ThreadedMonitor was configured to write an alert to the error log if a transaction took more than 1s to complete. See 6.2.4, “CICS TG monitoring exits” on page 202 for details on ThreadedMonitor.

Attention: Our workload was entirely sync-on-return transactions. Sync-on-return means that each individual ECI request flowed to the Gateway is an individual unit of work. CICS syncpoints the request before sending the response back to the Gateway. We therefore refer to each request as a transaction.

Warning signs

Users occasionally report delays. The descriptions they provide are not explicit enough to assist in problem diagnosis. There are many components involved in the system and it is unclear in which one we are seeing the delay.

Diagnosis

We looked at the average response times in Table 6-21.

Table 6-21 Average response times

Stat name	Value
GD_IAVRESP	207
CS_IAVRESP	209

These statistics do not show us anything useful because they are averages taken over a large number of requests.

Next, we looked at the output from the monitoring application (Example 6-7) to get information at a more granular level. The monitoring sample had written a number of alerts to the STDERR log.

Example 6-7 Alert from ThreadedMonitor

```
2007-10-09 08:22:32 !LRT-ALERT! [Program: CICDELY2] [TransId: null]
[Rc: OK(0)] [Start: Tue Oct 09 08:22:22 EDT 2007] [End: Tue Oct
09 08:22:32 EDT 2007] [Time: 10002ms]
2007-10-09 08:23:07 !LRT-ALERT! [Program: CICDELY2] [TransId: null]
[Rc: OK(0)] [Start: Tue Oct 09 08:22:56 EDT 2007] [End: Tue Oct
09 08:23:06 EDT 2007] [Time: 10001ms]
```

This indicated that one specific transaction, CICDELY2, was producing an abnormal delay. There were no alerts for any other transactions.

We looked for entries on the same time stamp of the detailed transaction output to get more information about the CICDELY2 transaction (Example 6-8).

Example 6-8 Detailed output from ThreadedMonitor

```
com.ibm.ctg.samples.requestexit.threadedmonitor.QueueMonitor:run called
with event = RequestEntry
Data[0000007C]: CtgCorrelator = 124
Data[0000007C]: RequestReceived (1191932542279) = Tue Oct 09 08:22:22
EDT 2007 offset = 0
Data[0000007C]: FlowType = EciSynconreturn
Data[0000007C]: Program = CICDELY2
```

```
Data[0000007C]: Location = /127.0.0.1
Data[0000007C]: Topology = Gateway
Data[0000007C]: WireSize = 114
Data[0000007C]: User correlator {empty}

com.ibm.ctg.samples.requestexit.threadedmonitor.QueueMonitor:run called
with event = ResponseExit
Data[0000007C]: CtgCorrelator = 124
Data[0000007C]: RequestReceived (1191932542279) = Tue Oct 09 08:22:22
EDT 2007 offset = 0
Data[0000007C]: RequestSent (1191932542279) = Tue Oct 09 08:22:22 EDT
2007 offset = 0
Data[0000007C]: ResponseReceived (1191932552281) = Tue Oct 09 08:22:32
EDT 2007 offset = 10002
Data[0000007C]: ResponseSent (1191932552281) = Tue Oct 09 08:22:32 EDT
2007 offset = 10002
Data[0000007C]: FlowType = EciSynconreturn
Data[0000007C]: Program = CICDELY2
Data[0000007C]: ClientLocation = /127.0.0.1
Data[0000007C]: Location = /127.0.0.1
Data[0000007C]: Topology = Gateway
Data[0000007C]: WireSize = 57
Data[0000007C]: CtgReturnCode = OK(0)
Data[0000007C]: CicsReturnCode = ECI_NO_ERROR(0)
Data[0000007C]: User correlator {empty}
Processing request 206
```

We looked at the offset values in ResponseExit to see which component of the system was causing the delay. The RequestSent offset was 0 ms and the ResponseReceived was 10002 ms. This indicates that the delay occurred within CICS.

We can also see that the transaction returned with successful return codes. The absence of a COMMAREA field in the reply indicates that the transaction did not return a COMMAREA.

Action taken

We were able to contact the CICS administrator about the problem and give explicit details on what program delayed the transaction and the return codes and data returned from the transaction.

Attention: In our example, we are using CICS TG on z/OS with EXCI. We assumed that delays are unlikely to be in the EXCI interface. If we were using the distributed CICS TG, or the IPIC protocol on the CICS TG on z/OS, it would be possible for delays to be in the network (between the CICS TG and CICS) in addition to within CICS.

6.4 Summary

A variety of system slow down scenarios can be successfully diagnosed and resolved using a combination of CICS TG statistics and the OMEGAMON XE and TEP tools. The new average response time statistics introduced in CICS TG V7.1 can help indicate where a performance constraint is occurring.

The monitoring exit samples that are introduced in CICS TG V7.1 provide the ability to diagnose more complex problems where the performance constraint is intermittent.



High availability with XA and OMEGAMON XE

In this chapter, we describe the different types of failure that can potentially happen when using the XA two-phase support to control global transactions initiated from WebSphere Application Server. We explore how OMEGAMON XE can help to quickly identify issues in comparison with the conventional information provided either by either the CICS Transaction Gateway (CICS TG) or CICS Transaction Server (CICS TS).

7.1 Scenario introduction

These are the components of our global scenario:

- ▶ WebSphere Application Server V6.1 on Linux for System z, which functions as the Transaction manager using the transactional capabilities of the CICS ECI XA resource adapter and the RRS support of the CICS Transaction Gateway on z/OS.
- ▶ An EJB application (CTGTesterCCIXA) deployed in WebSphere Application Server. For more details on this application, refer to Appendix A, “Sample J2EE application - CTGTesterCCIXA” on page 353.
- ▶ Two Gateway daemons running on z/OS and configured either:
 - As clones in port sharing mode and listening on the same TCP/IP port for the XA 1PC configuration
 - or
 - As stand-alone instances and listening on two different TCP/IP ports for the XA 2PC configuration
- ▶ Two CICS regions serving the ECI requests.
- ▶ The TCP/IP network.

In order to test our environment and check what happens in case of failure of one or more of the components listed previously, we needed to create and run workloads. In our scenario, we simulated heavy load on the servers (WebSphere Application Server and CICS) by using a specific tool HTTP test tool designed to load test functional behavior and measure performance. If you wish to reproduce the tests we describe in this chapter, you can chose any HTML based workload tool with which you are most familiar.

Note: Please refer to 4.4, “WebSphere Application Server configuration” on page 141 for a detailed explanation of our XA two-phase commit (2PC) and one-phase commit (1PC) scenarios and how to configure both WebSphere Application Server and the Java application for the different test scenarios.

7.2 High availability: XA 1PC configuration

Before proceeding with our description, it is probably worthwhile to briefly remind ourselves what the XA 1PC configuration means in terms of CICS TG and WebSphere Application Server customization:

- ▶ From WebSphere Application Server point of view, a 1PC scenario implies that only *one* managed resource is updated during the transaction. Therefore, we defined only one J2C Connection Factory representing one (virtual) CICS region.
- ▶ From the CICS TG point of view, this one Connection Factory was implemented as two cloned Gateway daemons (SCSCT711 and SCSCT714) listening on a shared TCP/IP port (2006), and each Gateway daemon sent requests to a dedicated CICS region (see Figure 7-1).

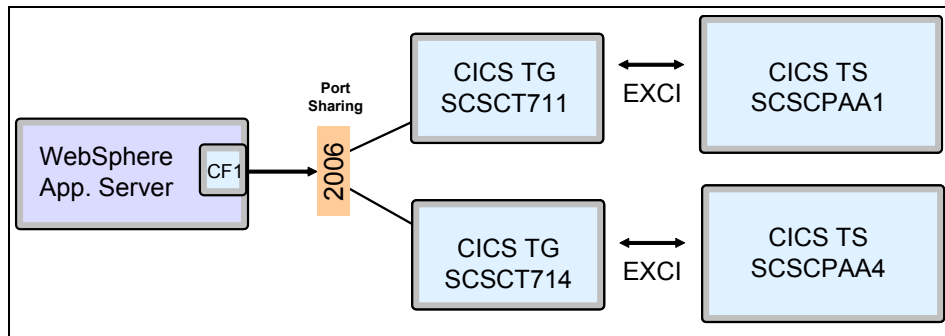


Figure 7-1 XA 1PC configuration

Initially, we used OMEGAMON XE under normal conditions to ascertain all the components were functioning normally. We then focused our attention on what occurs when one component of our global scenario either is not available or crashes.

This is the list of the different scenarios we analyzed and the tests we performed:

1. High availability when one Gateway daemon is unavailable.
2. High availability when one Gateway daemon crashes.
3. High availability when a CICS region crashes.
4. High availability when the network goes down.

7.2.1 Using OMEGAMON XE

The objective of this first test was to become familiar with the information OMEGAMON XE can provide when monitoring our environment under normal operating conditions when both the Gateway daemons and CICS regions were available.

Using our HTTP workload driver, we were able to simulate a large number of concurrent requests to our CTGTesterCCIXA application. For each request, the response we would expect from WebSphere Application Server was the HTML page shown in Figure 7-2.

Results	COMMAREA
Input 1:	
Output 1 using IBM037:	ëà&& "ë~'□'□□□□ë""\$"□\$TMCanâîgâæcæcæas(nccccccccccc
Output 1 using default encoding:	SCSCPAA1 08/10/07 13:30:39 CICSUSER D CSMI
Output 1 in HEX:	53 43 53 43 50 41 41 31 20 30 38 2F 31 30 2F 30 37 20 31 33 3A 33 30 3A 33 39 20 20 20 20 20 20 20 00 00 00 00 00 00 00
Input 2:	
Output 2 using IBM037:	ëà&& "ë~'□'□□□□ë""\$"□\$TMCanâîgâæcæcæas(nccccccccccc
Output 2 using default encoding:	SCSCPAA4 08/10/07 13:30:39 CICSUSER D CSMI
Output 2 in HEX:	53 43 53 43 50 41 41 34 20 30 38 2F 31 30 2F 30 37 20 31 33 3A 33 30 3A 33 39 20 20 20 20 20 20 20 00 00 00 00 00 00 00
Request time (ms):	560
Request succeeded.	

Figure 7-2 CTGTesterCCIXA normal response page

The page returns various parameters, including the APPLID from each of the two CICS regions that have served the ECI requests issued by the WebSphere application. Since the Gateway daemons are configured to use TCP/IP port sharing, it is not possible to know which Gateway daemon port sharing will send the requests to and so which CICS region will execute any given ECI call. In some cases, we can see two calls to the SCSCPAA1 region, in other cases, one call to both SCSCPAA1 and SCSCPAA4.

Thus, a given workload of 200 concurrent transactions to our CTGTesterCCIXA application will mean that 400 requests in total will be sent to the two CICS regions.

Before the run, the overall CICS TG situation shown by the TEP client for our two Gateway daemons is displayed in Figure 7-3 and Figure 7-4. These figures show that each Gateway daemon has 500 Connection Manager threads created, with none in use (allocated), and are thus ready for the workload.

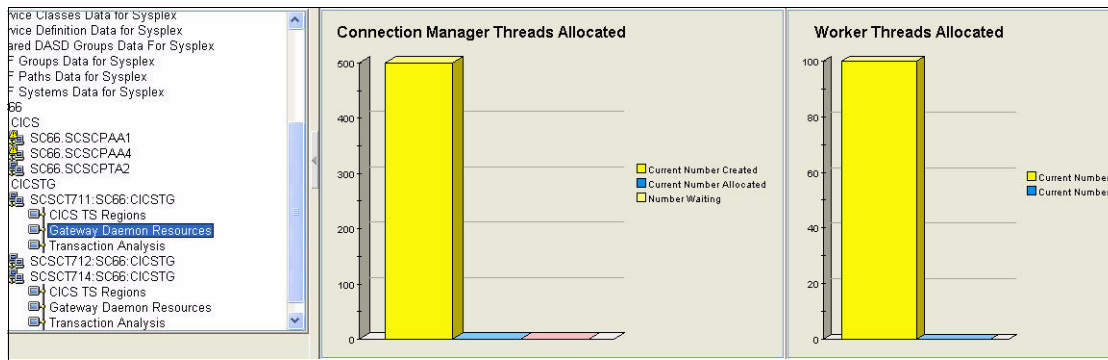


Figure 7-3 Gateway daemon SCST711 - before the run

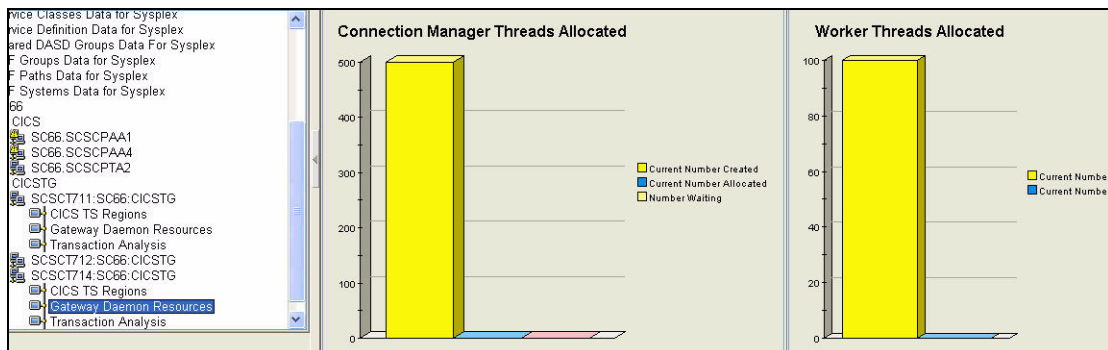


Figure 7-4 Gateway daemon SCST714 - before the run

After the run, we noticed the following differences:

1. In terms of Connection Manager threads, we saw that we had twelve threads allocated to SCST711 (Figure 7-5) and eight threads to SCST714 (Figure 7-6). This indicates that socket connections were now actually allocated from WebSphere to each Gateway daemon.

In terms of Worker threads, we saw that we had zero threads allocated in each Gateway daemon, as each gateway had finished its work.

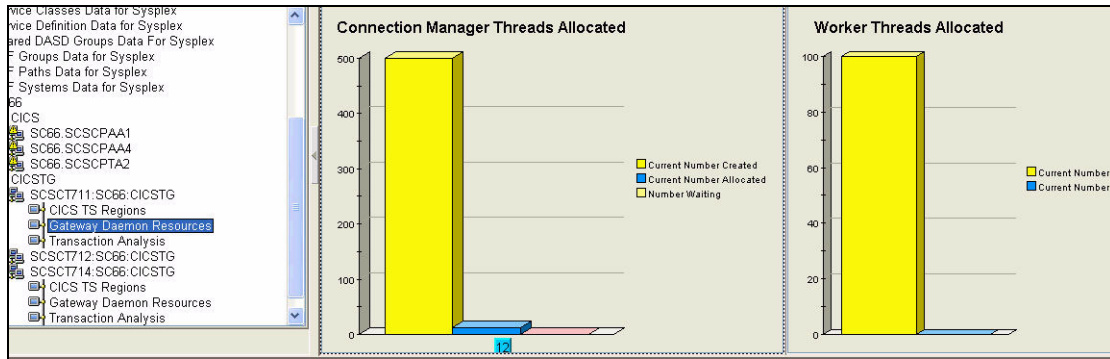


Figure 7-5 CICS TG SCST711 - after the run

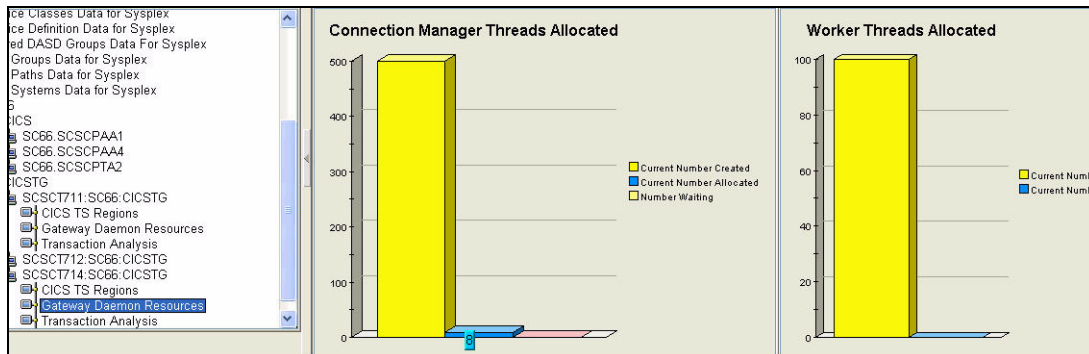


Figure 7-6 CICS TG SCST714 - after the run

2. Additionally, the OMEGAMON XE Transaction Analysis perspective showed us that in our tests TCP/IP port sharing did not evenly balance the incoming transactions across the cloned Gateway daemons. In Figure 7-7 on page 248 and Figure 7-8 on page 248, we can see that:
 - 258 XA transactions requested were committed by Gateway daemon SCSCT711.
 - 142 XA transactions requested were committed by Gateway daemon SCSCT714.
 - 258 XA transactions requested were committed by Gateway daemon SCSCT711.
 - 142 XA transactions requested were committed by Gateway daemon SCSCT714.

Note: Additionally, what the OMEGAMON XE figures also highlight is the fact that port sharing does not always evenly distribute socket connections across cloned server instances. In our case, sockets were allocated in the ratio of 12:8 (60%:40%) and this was reflected by the Transaction Analysis perspectives (Figure 7-38 on page 267).

The figures show that the 65% of the transactions are managed by SCSCT711, which has 60% of the allocated socket connection, and the remaining 35% are managed by SCSCT714, which has 40% of the socket connections.

The uneven distribution of work is a direct effect of the socket distribution because WebSphere has no precedence when it allocates connection handles from its connection pool. However, as you can see, thanks to OMEGAMON XE we can have an immediate view of what is happening within the systems, and with one glance can understand the system usage.

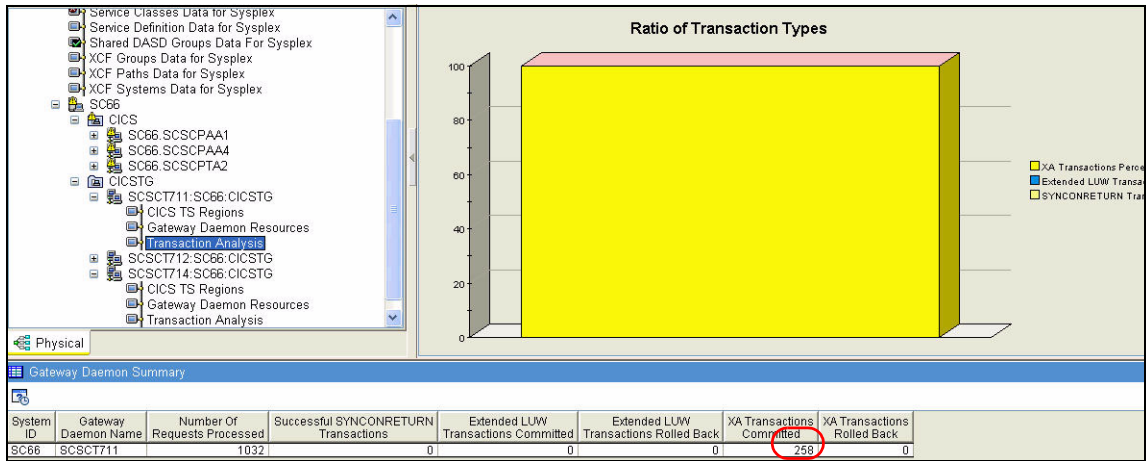


Figure 7-7 XA transactions committed on Gateway daemon SCSCT711

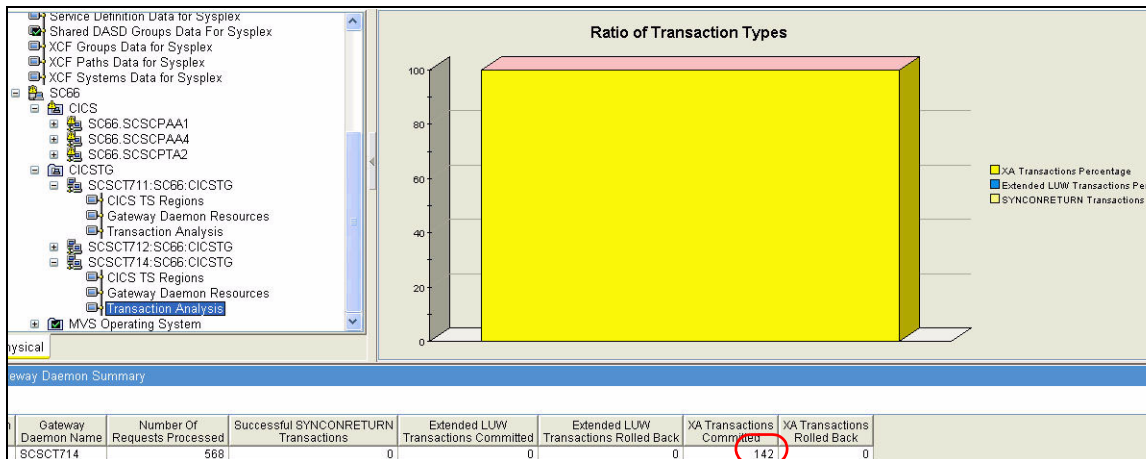


Figure 7-8 XA transactions committed on Gateway daemon SCSCT714

The numbers show that the 64.5% of the transactions are managed by SCSCT711 and the remaining 35.5% by SCSCT714. As you can see, thanks to the OMEGAMON XE features, we can have an immediate view of what is happening within the systems, and with one glance can understand if we are short of resources or not.

Note: We could obtain the same kind of information analyzing the CICS TG statistics, but this requires more effort and understanding of the CICS TG statistics.

For example, if we want to discover the current number of both the Connection Managers and the Worker Threads allocated by one of our Gateway daemons, we would have had to carry out the following sequence of operations:

1. Log on to the z/OS system and start SDSF.
2. Run the MVS command /F <jobname>,APPL=STATS,GS=CM:WT,ST=C:I to find the current and interval values for the Connection Manager and Worker thread statistics.
3. Look for the following values:
 - CM_CALLOC finds the current number of allocated Connection Manager threads.
 - CM_IALLOCHI finds the peak number of allocated Connection Manager threads in the interval.
 - WT_CALLOC finds the currently allocated worker threads.
 - WT_IALLOCHI finds the value of the peak number of allocated worker threads in the interval.

In addition, by taking a look at the OMEGAMON XE Summaries View (Figure 7-9) and refreshing the data (Refresh Now or F5), we can immediately retrieve the latest real time data from the monitored Gateway daemons.

CICS TS Regions Summary										
System ID	Gateway Daemon Name	EXCI Pipes Allocated	CICS TS Region Count	Requests Executed	EXCI Pipe Limit	EXCI Pipe Reallocations	EXCI Pipe Allocation Failures	Communication Failures	Free Pipes	EXCI NETNAME
SC66	SCSCT711	88	1	1711	100	0	0	1564	12	SCSCT711
Individual CICS TS Region Summaries										
System ID	CICS TS Region Jobname	CICS TS Region Applid	EXCI Pipes Allocated	Requests Executed	Requests Executed Per Minute	EXCI Pipe Allocate Failures	EXCI Pipe Allocation Failures Per Minute			
SC66	SCSCPAA1	SCSCPAA1	88	1711	0	0	0			

Figure 7-9 OMEGAMON XE Summaries Views

7.2.2 Gateway daemon unavailable

Next we wanted to analyze what happens in the case where one of the two Gateway daemons was not online. Since the Gateway daemon are clones listening on the same TCP/IP port, we did not expect to see any failures from the CICS transactions point of view, but simply to see all the traffic redirected to the only available Gateway and sent to its default CICS region (Figure 7-10).

Note: In our scenario, each Gateway daemon is associated with only one CICS region (its default region) and cannot send incoming requests to the other CICS, even if it is available.

Thus, in such a failure situation, the remaining Gateway daemon/CICS region pair must have enough capacity and free resources to serve all the incoming requests, if you do not want to experience a performance degradation.

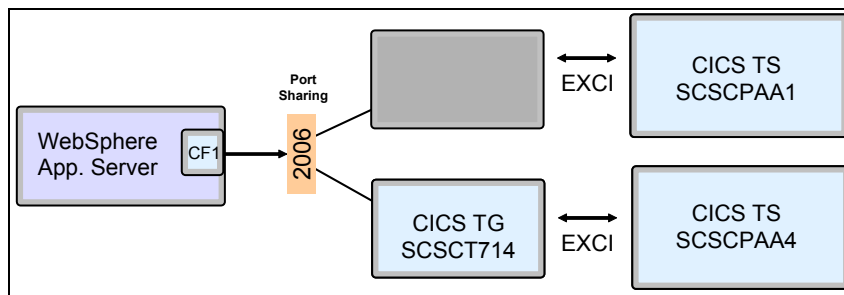


Figure 7-10 Gateway daemon SCST711 unavailable

The first thing we can immediately notice from the TEP client is which Gateway daemon is not online, since an unavailable Gateway daemon will be colored in grey instead of black. Figure 7-11 shows our example where the Gateway daemon SCST711 is down and SCST714 is still active.

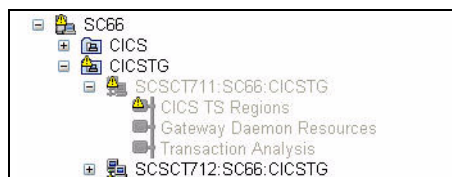


Figure 7-11 SCST711 is not active

Upon running our usual 200 concurrent requests workload, we noted the following information from the different OMEGAMON XE perspectives:

- 1. As expected, the whole workload (400 CICS requests) was managed by the single Gateway daemon SCST714 (see Figure 7-12).

Gateway Daemon Summary							
System ID	Gateway Daemon Name	Number Of Requests Processed	Successful SYNCONRETURN Transactions	Extended LUW Transactions Committed	Extended LUW Transactions Rolled Back	XA Transactions Committed	XA Transactions Rolled Back
SC66	SCST714	3370	166	0	0	400	0

Figure 7-12 XA transactions committed by SCST714

- 2. We did not get any abends from the CICS point of view, so we see 100% of the XA transactions committed (see Figure 7-13).

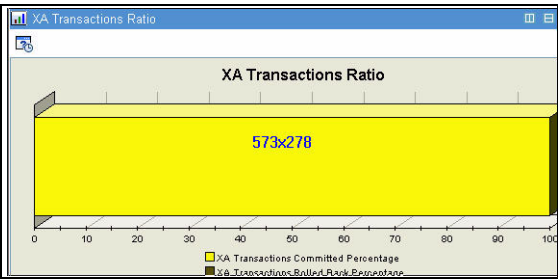


Figure 7-13 XA transaction ratio

- 3. The number of Connection Manager threads allocated is the sum of the two values (20) highlighted by the previous test (see Figure 7-14).

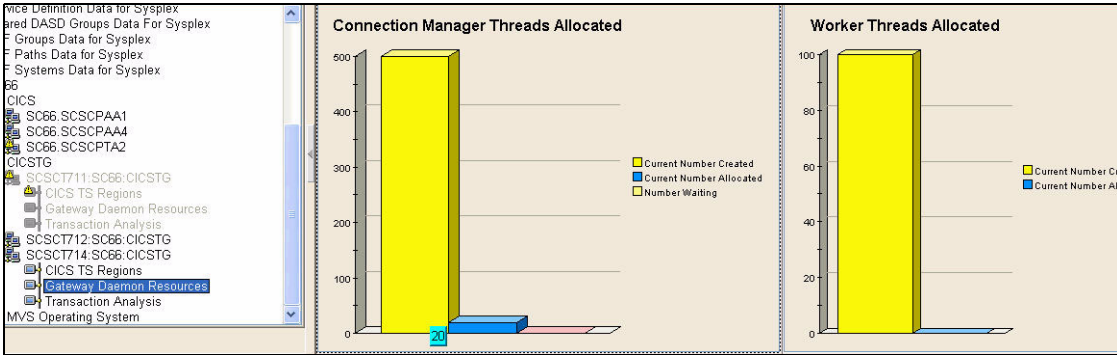


Figure 7-14 Connection Manager threads in use by SCST714

7.2.3 Gateway daemon failure

In this section, we will highlight what happens to our workload when a Gateway daemon suddenly fails (Figure 7-15).

Since our workload is using XA transaction, we should remember that WebSphere Application Server is functioning as the Transaction Manager and is managing the CICS transactions by means of the CICS ECI XA resource adapter and the Gateway daemons. If a Gateway daemon becomes unavailable, this will be detected by WebSphere Application Server either through the closure of the socket or during the prepare phase of the transaction. Either way, the transaction in CICS will be rolled back, and we are likely to see XA rollback requests sent to the CICS Transaction Gateway, until the problem is rectified.

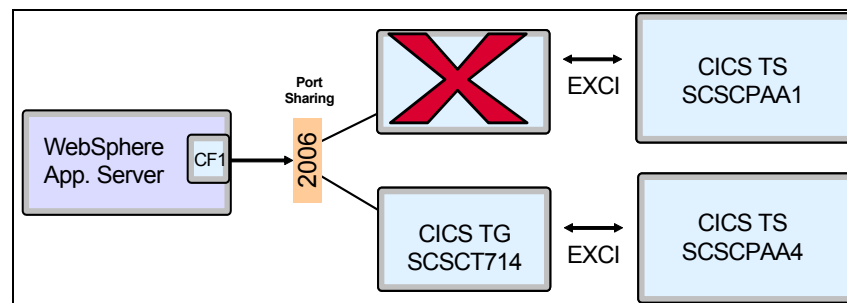


Figure 7-15 SCST711 fails

In our scenario, using TCP/IP port sharing, we expect to see the following:

1. WebSphere Application Server rolling back the in-flight transactions sent to the crashed Gateway daemon.
2. TCP/IP port sharing quickly detecting the problem and sending new requests to the remaining Gateway daemon (SCST714).

For this test, we need to simulate a workload longer than the usual 200 concurrent requests. This is because we must have the time to force the Gateway daemon to crash, analyze the situation using OMEGAMON XE, and restart the Gateway daemon.

This is the environment we used for our test:

- Workload Simulation Tool level: We decided to run a workload of 300 concurrent requests with an iteration of 120 times and with a think-time of four seconds between one transaction and the next one. By doing this task, we have a global workload of 36000 WebSphere XA transactions for a total of 72000 ECI requests/CICS transactions.

- WebSphere Application Server level: We had to modify the Connection pool properties parameters for the J2C connection factories used by our application. Please refer to Figure 7-16 for a comparison between our settings (on the left) and default values (on the right).

General Properties	General Properties
Scope cells:linux1Node01Cell:nodes:linux1Node01	Scope cells:linux1Node01Cell:nodes:linux1Node01
Connection timeout 10 seconds	Connection timeout 180 seconds
Maximum connections 500 connections	Maximum connections 10 connections
Minimum connections 50 connections	Minimum connections 1 connections
Reap time 60 seconds	Reap time 180 seconds
Unused timeout 60 seconds	Unused timeout 1800 seconds
Aged timeout 120 seconds	Aged timeout 0 seconds
Purge policy FailingConnectionOnly	Purge policy FailingConnectionOnly
Apply OK Reset Cancel	Apply OK Reset Cancel

Figure 7-16 Connection pool properties

The values we modified in the connection properties were as follows:

- Connection timeout (10s)
Specifies how long, in seconds, a getConnection() request by the J2EE application can wait if there are no connections available in the free pool and no new connections can be created. If a timeout occurs, a ConnectionWaitTimeoutException is thrown. We set this to 10 seconds in order to limit the querying of requests. By default, a request waits for 180 seconds.
- Maximum connections (500)
This is the maximum number of connections that can be created within the connection pool. We set this to 500 (the same as maxconnect in the Gateway) to ensure that this was not a limiting factor.
- Minimum connections (50)
This is the minimum number of connections that will be maintained in the connection pool after the pool manager maintenance thread has run. We set this to 50, the same as the number of WebSphere threads.

► Reap time (60)

This is the interval at which the pool manager maintenance thread will run. We reduced this to 60 seconds so that connections would be discarded quicker once they aged.

► Unused timeout (60)

This is the amount of time that connections will be allowed to remain idle before being eligible for closure by the pool manager maintenance thread. We set this to 60s so that connections would be closed down after one minute of inactivity.

► Aged timeout (120)

This is the amount of time that connections will be allowed to last before being eligible for closure by the pool manager maintenance thread. We set this to 120s to ensure that connections were recycled, allowing a restarted Gateway to be brought back online within a reasonable time.

Before launching the workload for this test, the TEP client displayed the window shown in Figure 7-17 for both Gateway daemons, indicating that there were no active transactions and no problems, that is, there were no red crosses next to the CICS TG line.

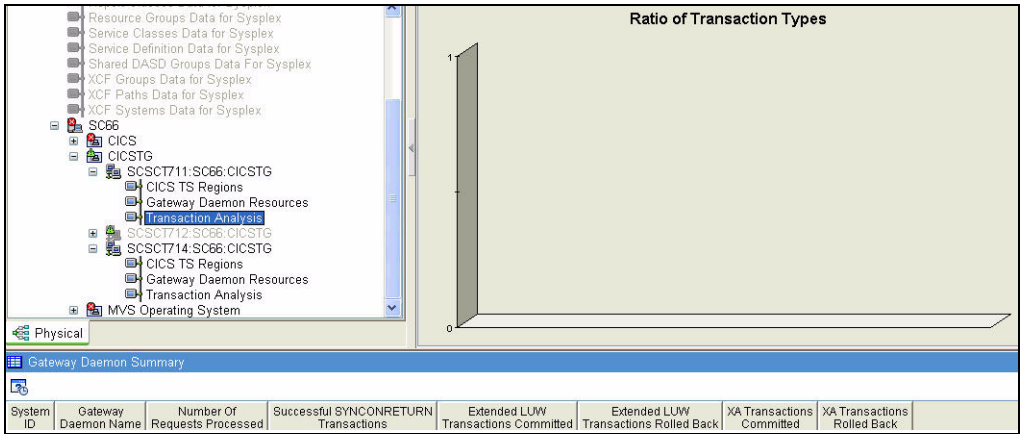


Figure 7-17 No active transactions for both SCSC711 and SCSC714

When we started the workload, we had the following situation showing on our TEP client console. It showed the numbers of XA committed transactions at the moment of inquiry (Figure 7-18 on page 255 and Figure 7-20 on page 256) and a total number of 100 (48 plus 52) Connection Manager Threads (Figure 7-19 on page 255 and Figure 7-21 on page 256) in use by the two Gateway daemons to manage the incoming requests.

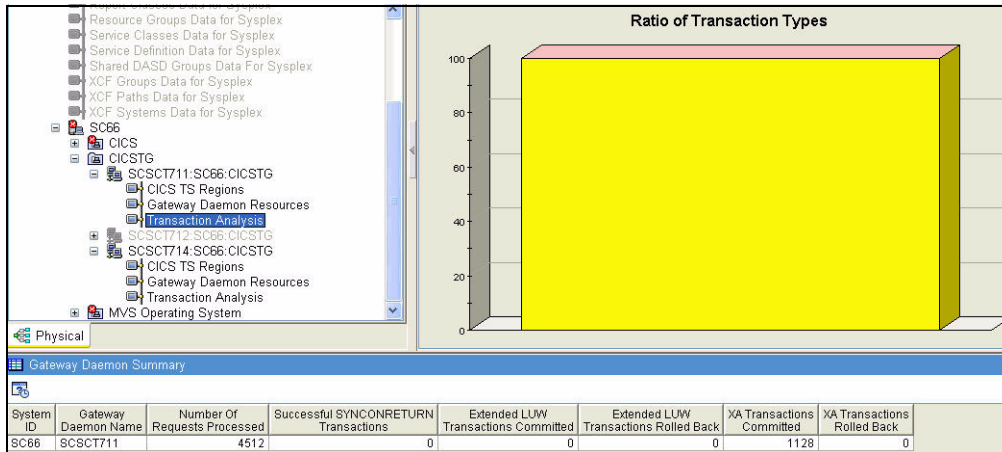


Figure 7-18 XA transactions committed by SCSC711

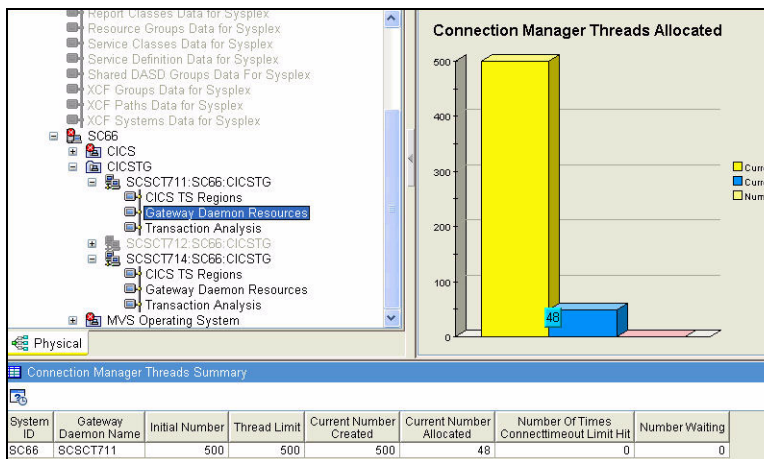


Figure 7-19 Connection Manager threads in use by SCSC711

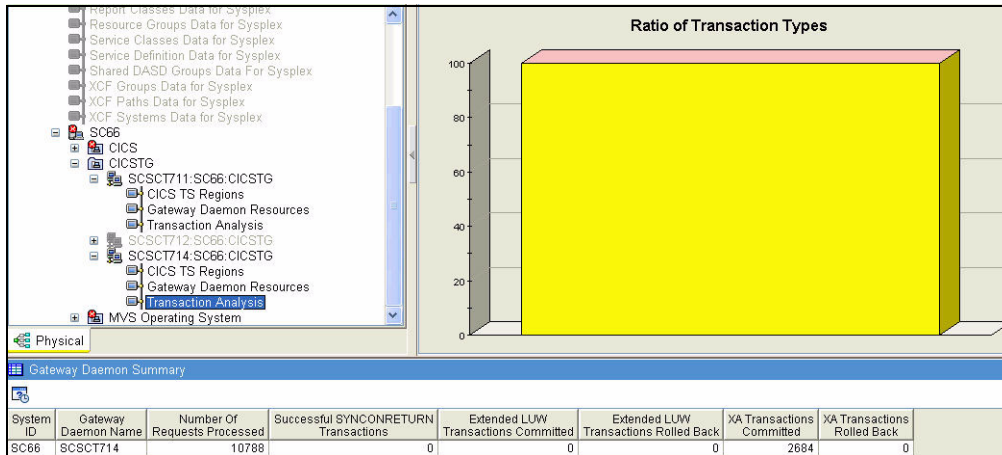


Figure 7-20 XA transactions committed by SCSC714

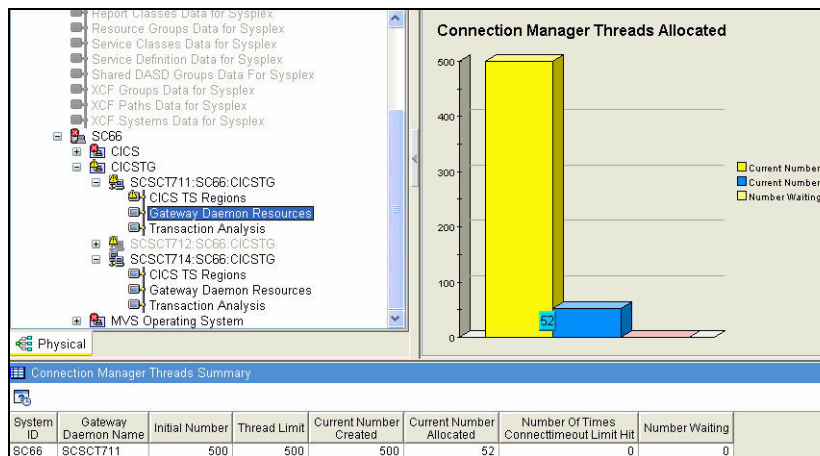


Figure 7-21 Connection Manager threads in use by SCSC714

After a while we cancelled the Gateway daemon SCSC711 job (Figure 7-22 on page 257).

COMMAND INPUT ==> /C SCSC711										SCROLL ==> CSR
PREFIX=SCSC* DEST=(ALL) OWNER=* FILTERS=2 SYSNAME=										
NP	JOBNAME	StepName	ProcStep	JobID	Owner	C Pos	DP	Real	Paging	SIO
SCSCT711	SCSCT711	CTG	STC02145	CTGUSER	NS	F6	53T	0.00	0.00	
SCSCPJA2	SCSCPJA2	CICS	STC15222	CICSTS	NS	F6	17T	0.00	0.00	
SCSCPAA4	SCSCPAA4	CICS	STC01512	CICSTS	NS	F8	75T	0.00	0.00	
SCSCPJA7	SCSCPJA7	CICS640	STC10473	CICSTS	NS	F6	4588	0.00	0.00	
SCSCT714	SCSCT714	CTG	STC02138	CTGUSER	NS	F6	54T	0.00	0.00	
SCSCPAA1	SCSCPAA1	CICS	STC01511	CICSTS	NS	F8	75T	0.00	0.00	

Figure 7-22 SCSC711 cancellation

Our HTTP workload driver tool immediately recorded the receipt of the application error page for some HTTP requests and OMEGAMON XE quickly detected the failure and colored the Gateway daemon SCSC711 grey (Figure 7-23).

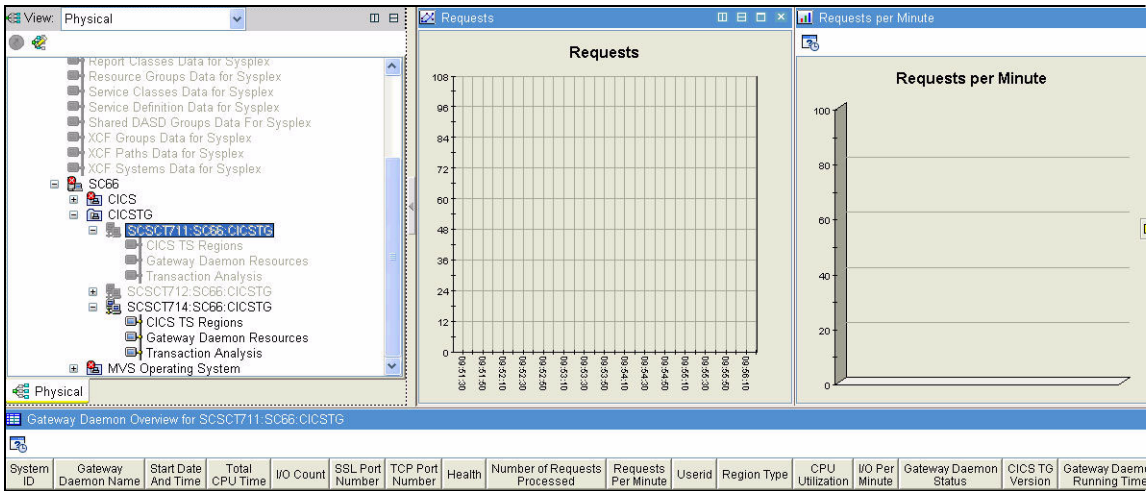


Figure 7-23 CICS TG SCSC711 unavailability

We now looked at SCSC714 (the only available Gateway daemon) in the TEP client (Figure 7-24 on page 258), and we saw that:

1. It had executed 12 XA rollbacks (the in-flight transactions as we expected).
2. It now had 100 Connection Manager threads allocated (Figure 7-25 on page 258).

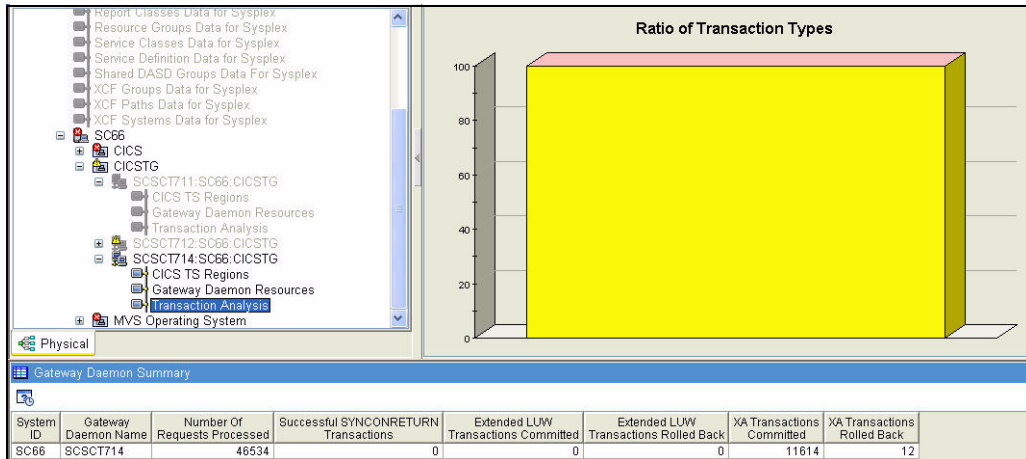


Figure 7-24 XA rollbacks by CICS TG SCSCT714

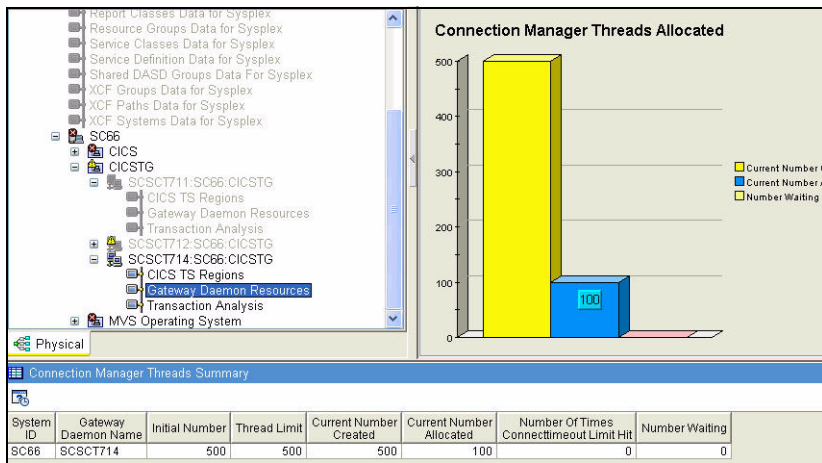


Figure 7-25 Connection Manager threads in use by CICS TG SCSCT714

Note: Although the crash happened to Gateway SCSCT711, the reason why SCSCT714 and not SCSCT711 performs the XA rollbacks is because when WebSphere Application Server asks for the execution of the rollbacks, they are managed by any available Gateway daemon listening on the shared TCP/IP port, and in our scenario, this is SCSCT714.

At this point, we restarted the Gateway daemon SCSCT711. In our scenario, it took several seconds for OMEGAMON XE to detect that the Gateway SCSCT711 was alive again and it took less than two minutes for the new

incoming requests to be directed to this Gateway using the shared port. This is shown by Figure 7-26 and Figure 7-27, which illustrate that SCST711 is working again and is using 52 Connection Manager threads (four more than those in use before its crash), while SCST714, which never stopped, is now serving a normal workload (Figure 7-28 on page 260).

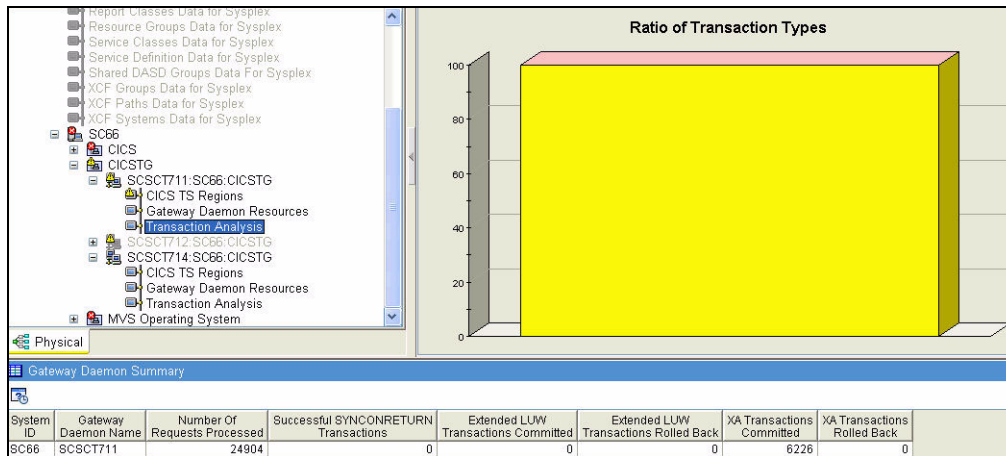


Figure 7-26 CICS TG SCST711 available again

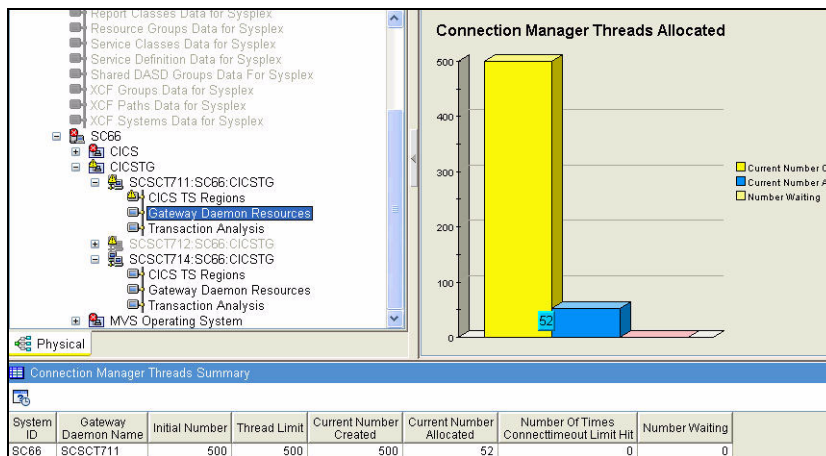


Figure 7-27 Connection Manager threads in use again by CICS TG SCST711

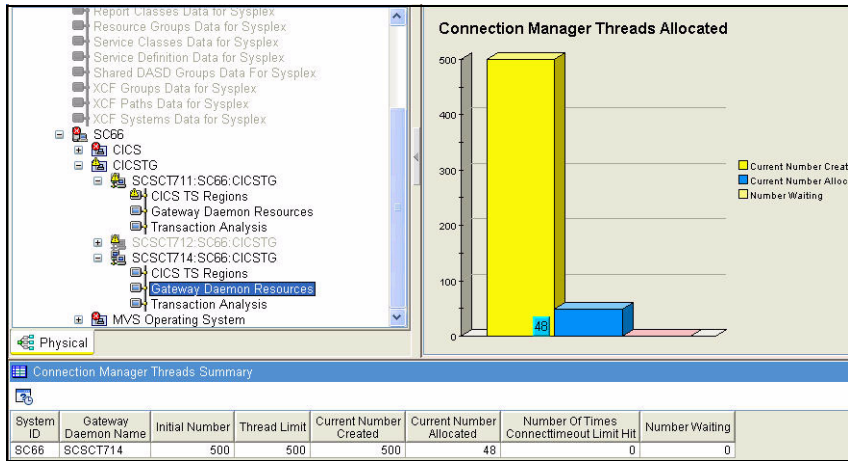


Figure 7-28 Connection Manager threads in use by CICS TG SCSC714

Important: When the crashed Gateway daemon restarts, you may see some XA rollbacks if the Gateway is restarted relatively quickly. In such a situation, because the WebSphere transaction manager replays rollbacks until they are acknowledged, it may be possible that some rollbacks will be processed by the restarted Gateway daemon.

7.2.4 CICS region failure

In this section, we will describe the result of our observations in the case of the failure of one of the two CICS regions in our environment (Figure 7-29).

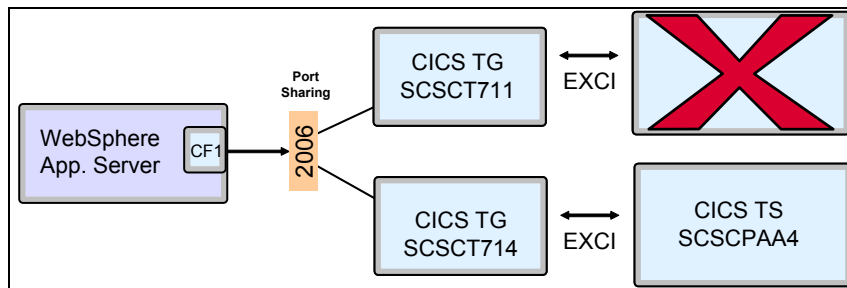


Figure 7-29 CICS region failure

Storm drain scenario

The failure of a CICS region being used by a set of cloned Gateway daemons using TCP/IP port sharing can cause a *storm drain* situation, where new TCP/IP connection requests continue to be sent to the Gateway daemon serving the failed CICS region. This is because the CICS failure is not visible to the TCP/IP load balancer, which can only view the number of connections and the efficiency of TCP/IP socket processing. There are essentially two methods that can be used to address this issue, either the *DFHXCURM* exit or the CICS TG *health reporting* function.

EXCI DFHXCURM user exit

The DFHXCURM user exit works at the EXCI level and allows EXCI allocation and retryable errors to be retried. It can be used to remove the affinity of an EXCI request from a given CICS region by dynamically modifying the APPLID in the initial EXCI allocation request. It can also be used to provide limited workload balancing of EXCI requests across a sysplex.

The sequence of events that happens inside the Gateway daemon is:

1. The CICS TG will try to open a pipe to the bad CICS region and this will fail with a retryable error.
2. The CICS TG will then try to allocate a new pipe to that CICS region (this can occur up to five times for each retryable failure).
3. This will invoke the DFHXCURM, allowing the APPLID of the CICS region to be modified before it is sent.
4. The allocation will now be sent to the good CICS region, and further EXCI requests using this EXCI pipe should succeed.

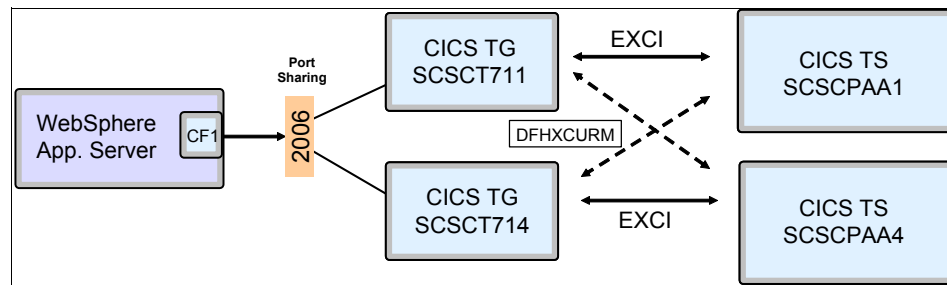


Figure 7-30 DFHXCURM function

There are the following issues with the DFHXCURM exit:

- ▶ Since the CICS TG caches EXCI pipes, it generally only reissues an allocation when there is a failure. However, since the DFHXCURM exit only runs on a pipe allocation or retryable error, it will not be invoked when a failed CICS region is restarted. Thus, a restarted CICS region will remain dormant until further pipes are allocated.
- ▶ The CICS TG per-server EXCI statistics (resource group CSX) will become invalid if a DFHXCURM is used, since pipes may be re-allocated to a different server.

CICS TG health reporting function

The CICS TG V7.1 integrates with WLM to allow exploitation of server-specific recommendations when using TCP/IP port sharing or Sysplex Distributor. Health is reported by the CICS TG as a percentage value from 100 to zero reflecting the number of ECI request failures in a configurable interval. If all requests fail in the interval, the health is reported as zero and WLM prevents any further TCP/IP connections being sent to a given server. At this point, all new requests will be routed to the other CICS TG and hence a new CICS region (see Figure 7-31 for more details).

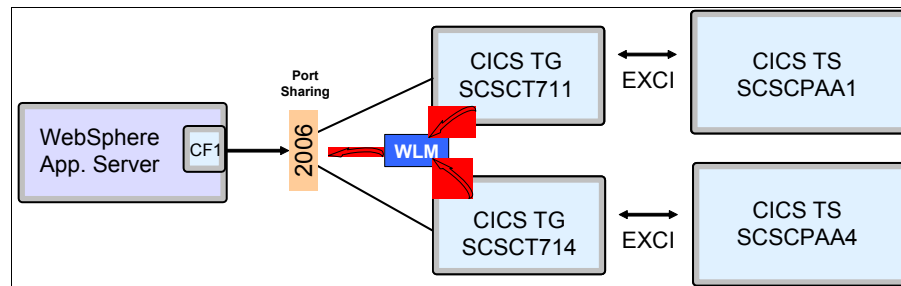


Figure 7-31 WLM health reporting

In CICS TG V7.0, the health reporting interval was fixed at 60 seconds. Customers complained that this was too slow, so in Version 7.1 a configurable interval was introduced, which can be set using the *healthinterval* parameter in the ctg.ini file, allowing much more rapid reporting of server failures to WLM. In addition, the SYSPLEXWLMPOLL parameter in TCP/IP can also be used to determine the frequency at which TCP/IP queries WLM to discover the health of systems. You may want to set this in conjunction with the healthinterval parameter to enable rapid discover of low health. For more details, refer to the following URL:

<http://www.ibm.com/support/docview.wss?uid=isg1PK24752>

The health is calculated based on the ECI request API return codes, and is the ratio between unsuccessful requests and successful requests that occurred during the health interval. Bad requests are classified as:

- ▶ ECI_ERR_NO_CICS
- ▶ ECI_ERR_RESOURCE_SHORTAGE
- ▶ ECI_ERR_SYSTEM_ERROR

These unsuccessful requests are also flagged as fatal connection errors by the CICS ECI resource adapter. This information is used by the WebSphere pool manager to purge failing connections from the pool, which will also result in the closure of the connection that had been in use at the time. New requests will then use either existing valid connections left in the pool, or will initiate new connections to be established with the Gateway daemon.

If health ever reaches zero, a Gateway will become dormant because it will not receive any new TCP/IP connections based on the WLM recommendations. Thus, even when the failed CICS region is brought back online, health will stay at 0 and not increase again. To get around this problem, the CICS TG provides a manual switch to set health back to 100. Run the following MVS modify command, which needs to be issued manually or through an automation feature:

```
/F JOB,APPL=HEALTH,RESET
```

Testing the use of the DFHXCURM user exit

A sample DFHXCURM is provided with CICS TS in the SDFHSAMP library. We created a new version suitable for our environment and assembled and linked it into the CICSTS32.CICS.SDFHEXCI library used in our Gateway daemon STEPLIB and restarted our Gateway daemon. For details on how to obtain a copy of our modified DFHXCURM, refer to Appendix D, “Additional material” on page 379.

For this type of test, we adopted the same Workload Tool configuration that we used for the run with the CICS TG crash, so we had 300 concurrent requests, an iteration of 120 times, and think-time of four seconds between one transaction and the other. In total, we had a global workload of 36000 WebSphere transactions corresponding to 72000 ECI requests/CICS XA transactions. The only difference is that, this time, we decided to use the specific DANL mirror transaction, instead of the default CSMI, because we needed to be able to identify our transactions when we looked at the OMEGAMON XE Transaction Analysis Perspective.

We initiated the HTTP workload and then, using the TEP client, we could see that both the CICS regions were up and running (SCSCPAA1 and SCSCPAA4 in black) and both the Gateway daemons were working properly (SCSCT711 and SCSCT714 shown in black). Figure 7-32 and Figure 7-33 show the percentage of XA transactions committed and rolled back after each time the Gateway daemons SCSCT711 and SCSCT714 were pinged.

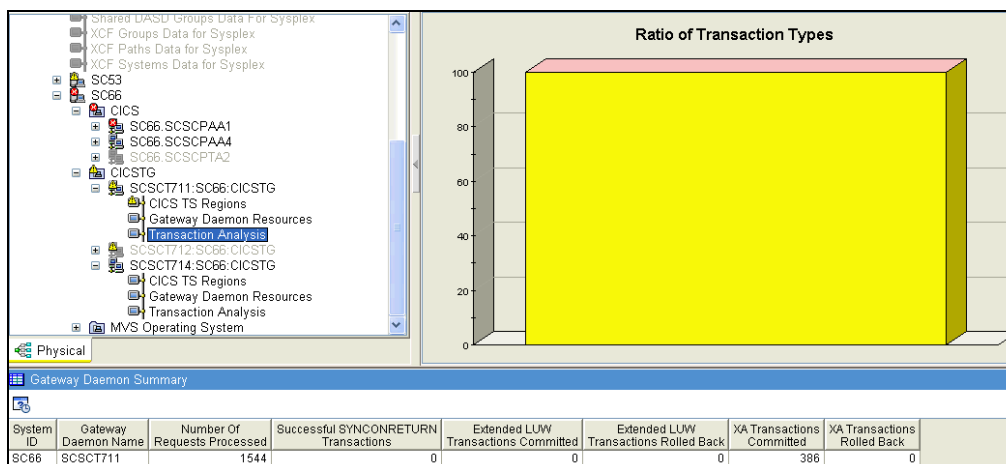


Figure 7-32 XA transactions status for SCSCT711

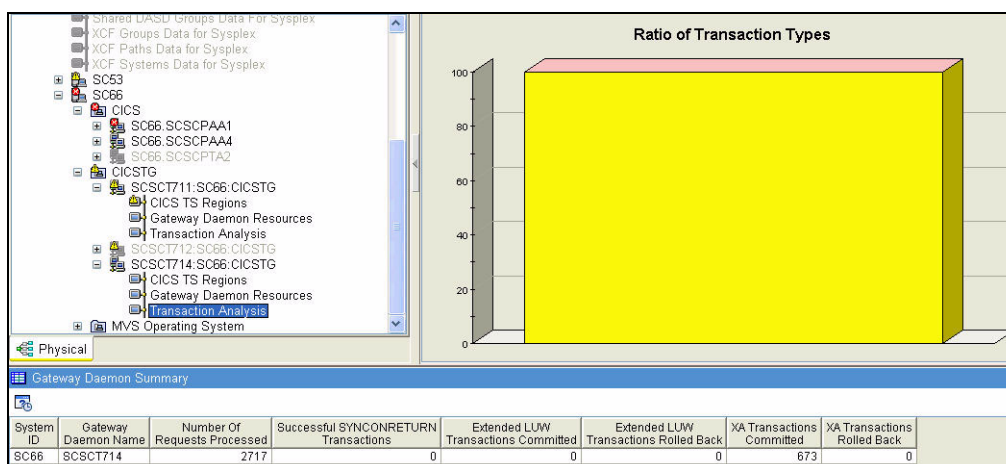


Figure 7-33 XA transaction status for SCSCT714

We decided to simulate a real CICS crash by cancelling the CICS region SCSCPAA1 region by using the MVS cancel command, as shown in Figure 7-34 on page 265.

```
COMMAND INPUT ==> /C SCSCPA1                                SCROLL ==> CSR
PREFIX=SCSC* DEST=(ALL) OWNER=* FILTERS=2 SYSNAME=
NP  JOBNAME  StepName ProcStep JobID   Owner   C Pos DP Real Paging   SIO
SCSCT711 SCSCT711 CTG      STC02145 CTGUSER   NS   F6 53T   0.00 0.00
      SCSCPJA2 SCSCPJA2 CICS      STC15222 CICTS      NS   F6 17T   0.00 0.00
SCSCPA4 SCSCPA4 CICS      STC01512 CICTS      NS   F8 75T   0.00 0.00
      SCSCPJA7 SCSCPJA7 CICS640  STC10473 CICTS      NS   F6 4588 0.00 0.00
SCSCT714 SCSCT714 CTG      STC02138 CTGUSER   NS   F6 54T   0.00 0.00
      SCSCT710 SCSCT710 MASTER  STC27460 CTGUSER   LO   FF 483   0.00 0.00
SCSCPA1 SCSCPA1 CICS      STC01511 CICTS      NS   F8 75T   0.00 0.00
```

Figure 7-34 Cancelling SCSCPA1

Once again, OMEGAMON XE was able to immediately detect the failure, as indicated by the red bar in CICS TS Region Jobname field in the TEP client console. (Figure 7-35).

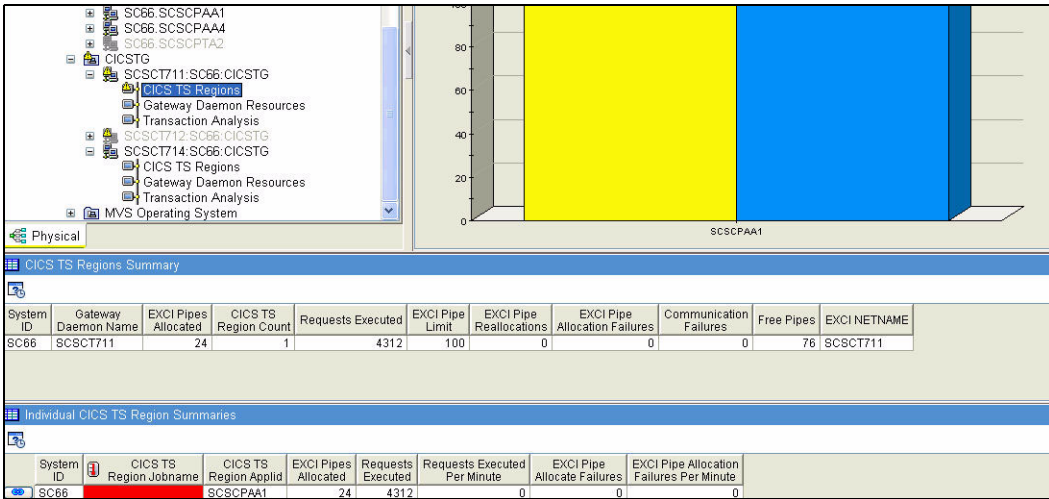


Figure 7-35 CICS region SCSCPA1 unavailable

Our DFHXCURM was designed to handle CICS region failure and therefore we did not expect to have close to zero WebSphere or CICS exceptions, so we used OMEGAMON to check this was the case. The following TEP client views (Figure 7-36 and Figure 7-37) show that both Gateway daemons have no XA rollbacks recorded after the CICS region SCSCPA1 had crashed.

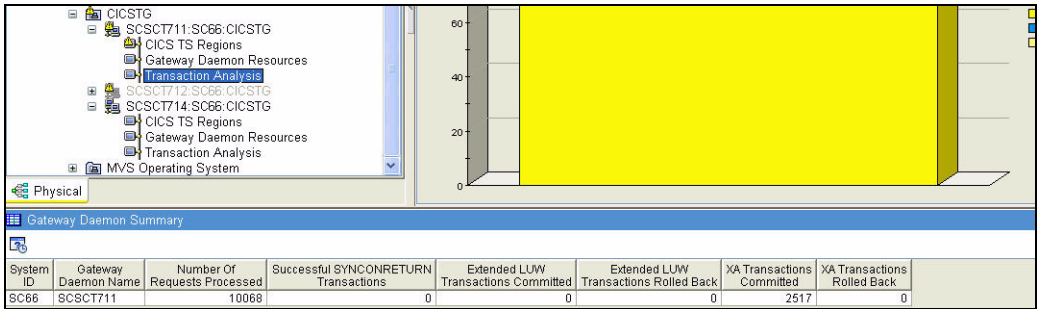


Figure 7-36 SCSC711 - no rollbacks

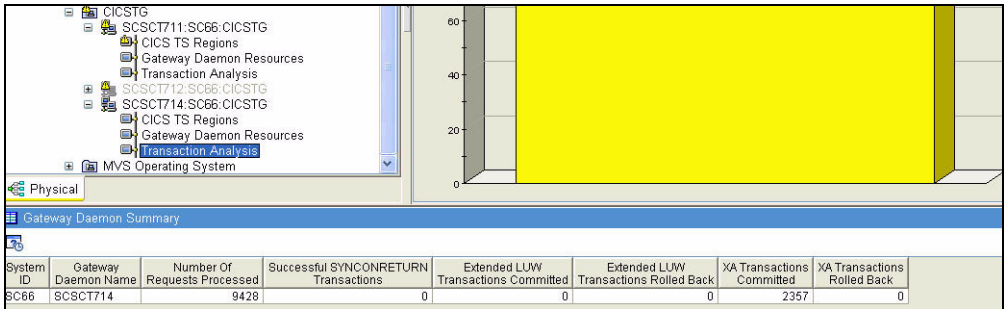


Figure 7-37 SCSC714 - no rollbacks

In the Transaction Analysis Perspective for CICS SCSCPA4 (Figure 7-38 on page 267), we can see our DANL mirror transaction in the first line, indicating that this CICS region is serving our ECI requests. We used the Dynamic Workspace Linking (DWL) feature of OMEGAMON XE for CICS TG to view the target Transaction Analysis workspace in the OMEGAMON XE for the CICS product.

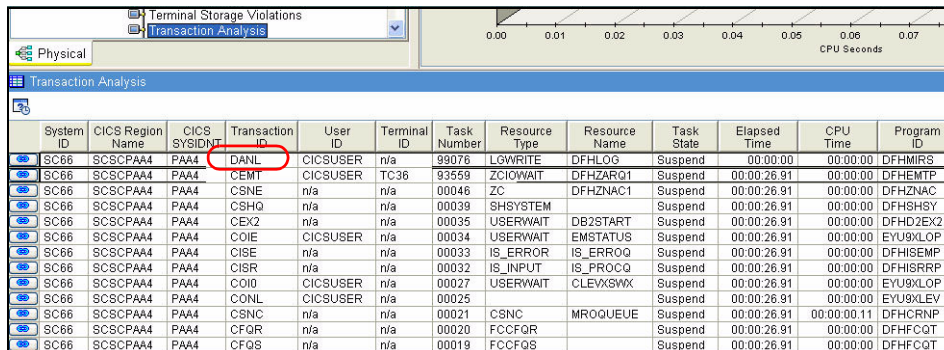


Figure 7-38 Transaction Analysis Perspective - DANL transaction

Once we restarted the CICS region SCSCPA1, nothing changed in the OMEGAMON XE values, which confirmed that CICS region SCSC711 will not be used after the failure, due to the fact that all EXCI pipes have been allocated to the remaining CICS region SCSCPA4. Another confirmation of this behavior was the fact that when we logged on to CICS SCSCPA1 and ran the CEMT transaction to inquire about the status of the CICS program ECIPROG, we noticed that the Use parameter was 0 and never increased in value, confirming that the CICS program was never called after the restart.

The last thing to check is the OMEGAMON XE Gateway Daemon Resources perspective in order to verify that both Gateway daemons are receiving the requests by the shared TCP/IP port. This was confirmed by the OMEGAMON XE Connection Manager values of 500 shown in Figure 7-38 and Figure 7-39 for each Gateway daemon.

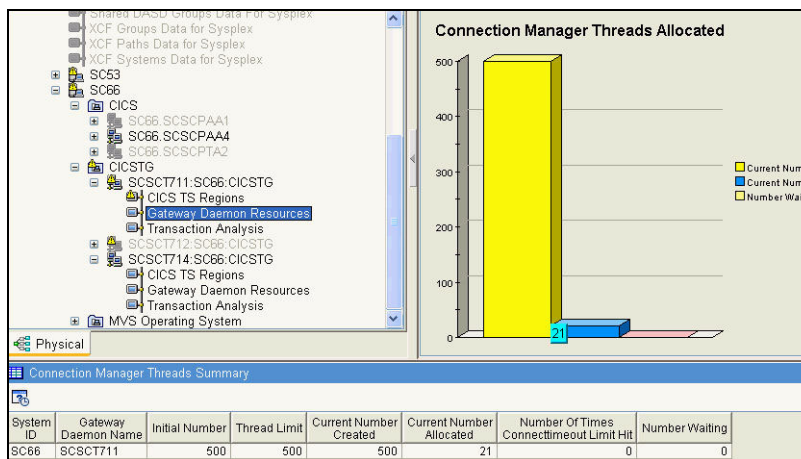


Figure 7-39 Connection Manager threads - SCSC711

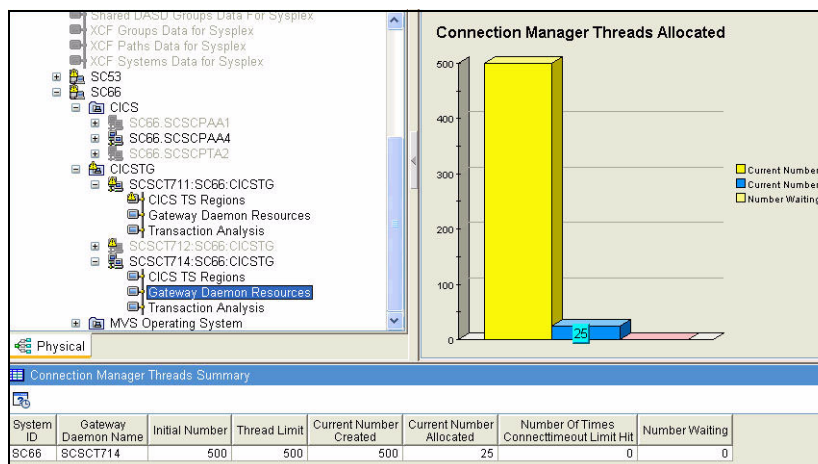


Figure 7-40 Connection Manager threads - SC66.CICSTG

Testing the use of the CICS TG health reporting

In this test, we highlight how we used CICS TG health reporting to accommodate a CICS region failure, and how to exploit the CICS TG health reset command and the new healthinterval parameter provided with CICS TG V7.1.

When a CICS region fails, the incoming ECI requests will be unsuccessful, which will cause the health to start to decrease from 100 to a lower value, depending on the ratio between the unsuccessful and successful requests. If all the requests fail, then very quickly the health will be reported as zero (depending on the healthinterval parameter) causing the Gateway daemon to be bypassed by TCP/IP port sharing for all new connections. Only when the health rises to a value greater than 0 will TCP/IP port sharing start again to insert that Gateway daemon into the normal distribution.

To test this situation, we used the same workload described in “Testing the use of the DFHXCURM user exit” on page 263. The workload was running correctly and verified using the TEP client. To initiate the failure, we canceled the CICS region SC66.CICSTG, and then we show what happened by displaying a different OMEGAMON XE TEP client view.

Having cancelled the CICS region SC66.CICSTG, we see that the CICS region SC66.CICSTG is now unavailable because no activity is displayed in the overall CICS region view (Figure 7-41 on page 269).

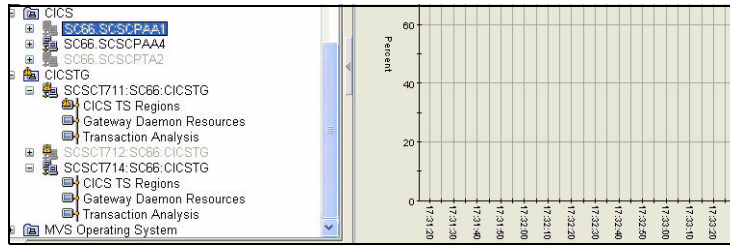


Figure 7-41 CICS TS SCSCPA1 down - no activity

This produced the following alert in the TEP client informing us about the problem (Figure 7-42).

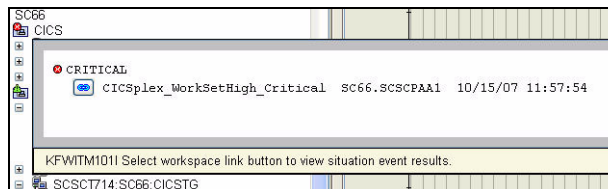


Figure 7-42 OMEGAMON XE alert about CICS stability

The same information is displayed in the CICS TS Regions view for the Gateway daemon (Figure 7-43).

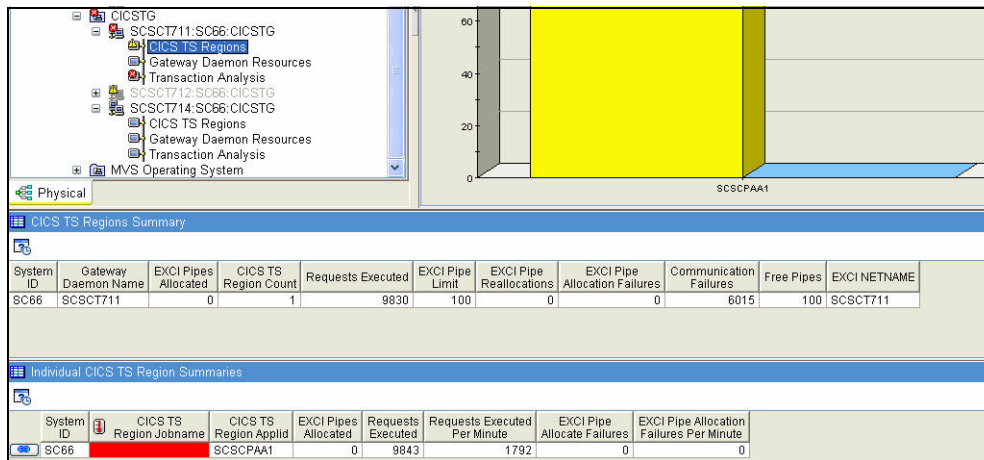


Figure 7-43 CICS SCSCPA1 - unavailable

The following sequence of OMEGAMON XE screen captures (Figure 7-44 and Figure 7-45) shows the CICS TG SCST711 health progressively decreasing from 100 to 30 and then 0.

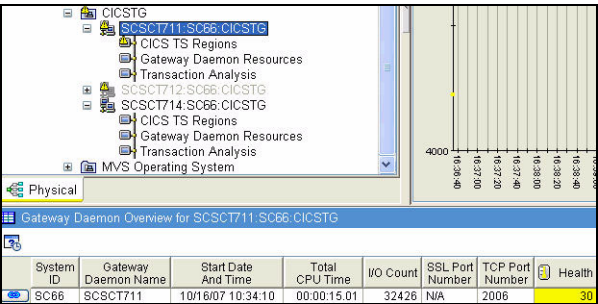


Figure 7-44 CICS TG SCST711 - health 30

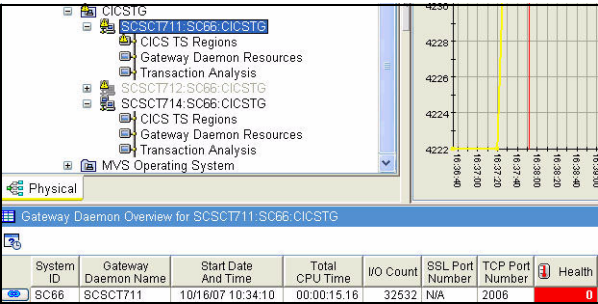


Figure 7-45 SCST711 - health zero

In the time frame between the CICS failure and the moment when the health of its associated CICS TG is set to 0, the users will experience a certain number of ECI call failures that will cause WebSphere to send an XA rollback request to the associated Gateway daemon. This number (114) is displayed in our OMEGAMON XE Transaction Analysis view (Figure 7-46 on page 271).

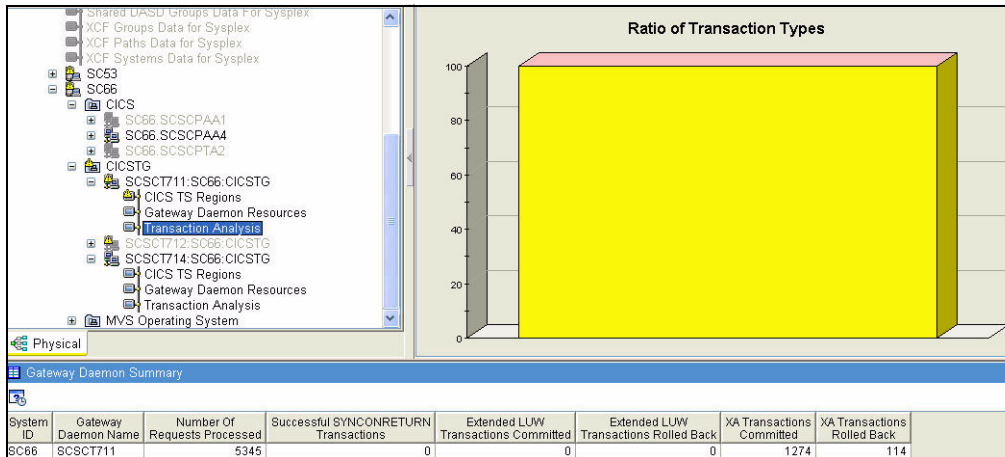


Figure 7-46 XA rollbacks in CICS TG SCSCT711 due to SCSCPA1 unavailability

We now see all the incoming traffic managed by the Gateway daemon SCSCT714. With the health now at zero, the Gateway daemon SCSCT711 will not receive any new TCP/IP connections, even when CICS SCSCPA1 is up and running again, unless the health is reset either manually or automatically back to 100. This is illustrated in Figure 7-47 and Figure 7-48 on page 272.

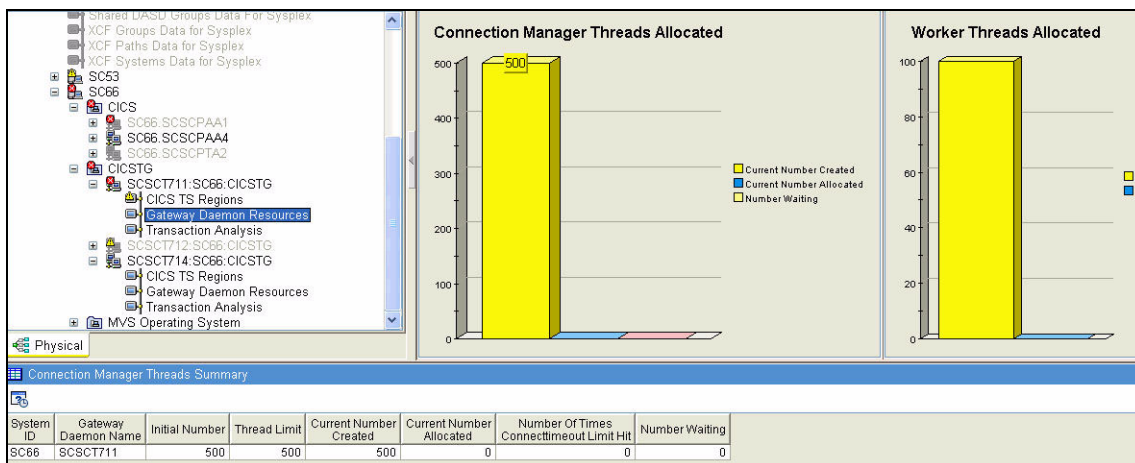


Figure 7-47 SCSCT711 not in use

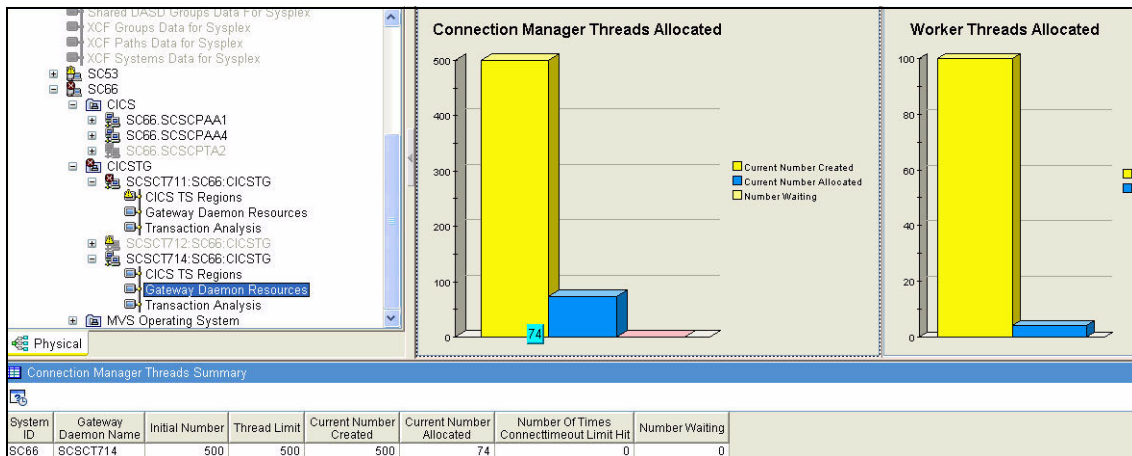


Figure 7-48 All incoming traffic managed by CICS TG SC66

At this point, we restarted CICS region SC66PAA1 and OMEGAMON XE quickly detected the change in status. When the startup procedure had completed, we checked if Gateway daemon SC66T711, which was associated with the restarted CICS region, was now in use; as expected, it was not, so we proceeded with resetting the health of SC66T711 using the MVS modify command to reset the health, as shown in Figure 7-49.

COMMAND INPUT ==> /F SC66T711,APPL=HEALTH,RESET											
PREFIX=SC66* DEST=(ALL) OWNER=* FILTERS=2 SYSNAME=											
NP	JOBNAME	StepName	ProcStep	JobID	Owner	C	Pos	DP	Real	Paging	SIO
	SC66T711	SC66T711	CTG	STC02145	CTGUSER	NS	F6	53T	0.00	0.00	
	SC66PJA2	SC66PJA2	CICS	STC15222	CICSTS	NS	F6	17T	0.00	0.00	
	SC66PAA4	SC66PAA4	CICS	STC01512	CICSTS	NS	F8	75T	0.00	0.00	
	SC66PJA7	SC66PJA7	CICS640	STC10473	CICSTS	NS	F6	4588	0.00	0.00	
	SC66T714	SC66T714	CTG	STC02138	CTGUSER	NS	F6	54T	0.00	0.00	
	SC66T710	SC66T710	MASTER	STC27460	CTGUSER	LO	FF	483	0.00	0.00	
	SC66PAA1	SC66PAA1	CICS	STC01511	CICSTS	NS	F8	75T	0.00	0.00	

Figure 7-49 CICS TG health reset

Immediately, OMEGAMON XE displayed the reset health and confirmed the use of SC66T711 by TCP/IP port sharing (Figure 7-50 on page 273 and Figure 7-51 on page 273).

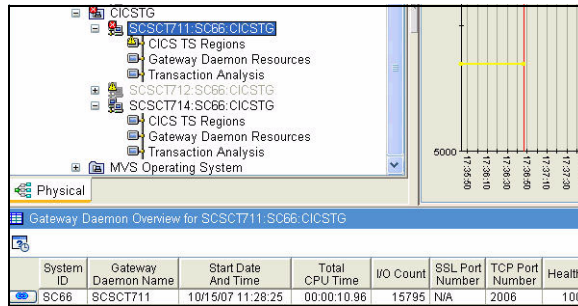


Figure 7-50 CICS TG SC SCT711 - health 100

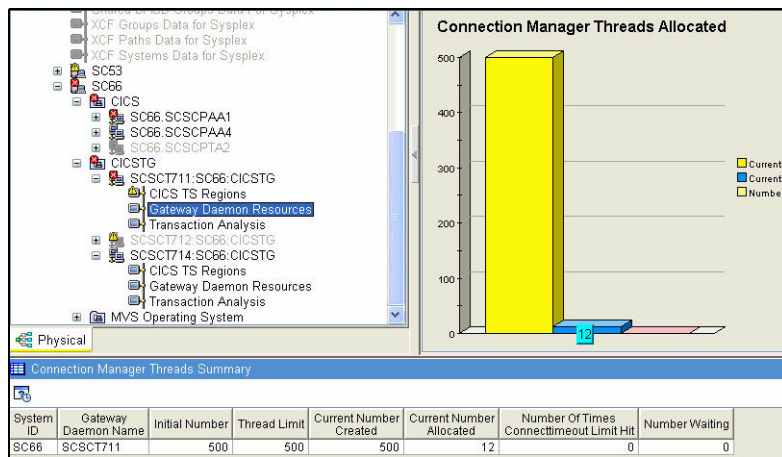


Figure 7-51 SC SCT711 - Connection Managers in use

A useful aspect that should be mentioned is that we can configure OMEGAMON XE to automatically change the Gateway daemon health once the restart of its associated CICS region has been completed. The OMEGAMON XE customization is quite simple, as outlined in the following sequence:

1. Open the OMEGAMON XE editor by either clicking the icon or pressing Ctrl+E, then select the **Monitored Application** to create the Situation for a CICS TG (Figure 7-52), and click the **Create new Situation** icon.

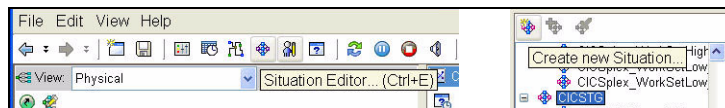
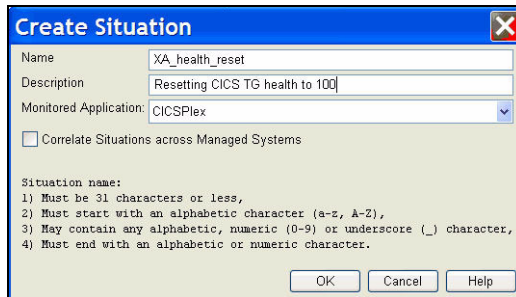


Figure 7-52 Situation Editor

2. Complete all the fields in the Create Situation window and press **OK**, as shown in Figure 7-53.



The 'Create Situation' dialog box has a blue title bar with the text 'Create Situation' and a close button. It contains several input fields: 'Name' with the value 'XA_health_reset', 'Description' with the value 'Resetting CICS TG health to 100', and 'Monitored Application' with a dropdown menu showing 'CICSplex'. Below these is an unchecked checkbox labeled 'Correlate Situations across Managed Systems'. At the bottom, there is a section titled 'Situation name:' followed by four numbered rules: 1) Must be 31 characters or less, 2) Must start with an alphabetic character (a-z, A-Z), 3) May contain any alphabetic, numeric (0-9) or underscore (_) character, and 4) Must end with an alphabetic or numeric character. At the very bottom are three buttons: 'OK', 'Cancel', and 'Help'.

Figure 7-53 Create Situation window

3. A new window appears. Here you select the conditions OMEGAMON XE has to check for our environment. In this situation, there are two conditions to identify:
 - a. CICS TG health
 - b. CICS TS Region Jobname

You have to choose **Health** (Figure 7-54 on page 275), which belongs to the CICSTG Region Overview and then, by pressing the **Add Conditions** tab (Figure 7-55 on page 275), select the second criteria **CICS TS Region Jobname** (belonging to CICSTG CICS TS Region Details), as shown in Figure 7-56 on page 276.

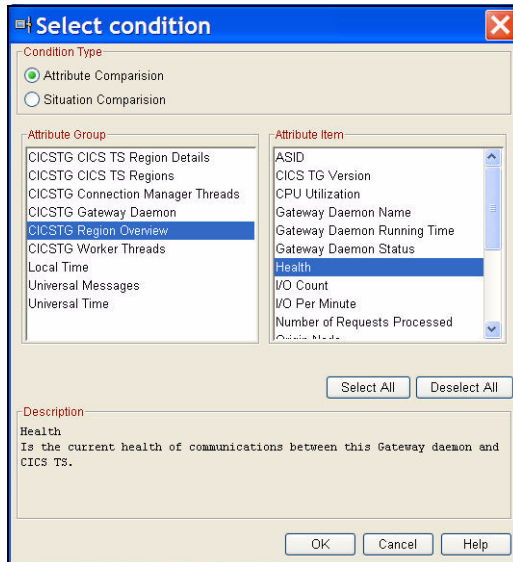


Figure 7-54 Health condition

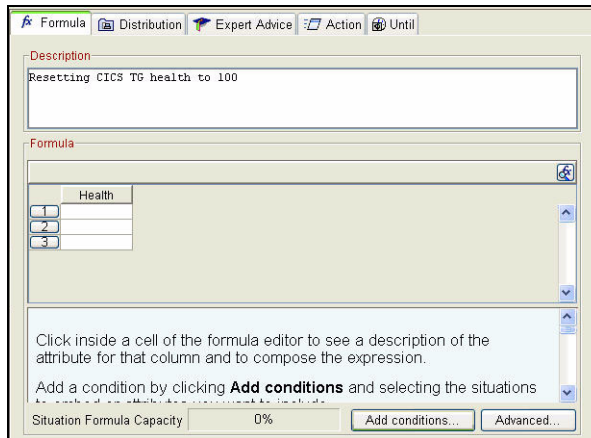


Figure 7-55 Add Conditions tab

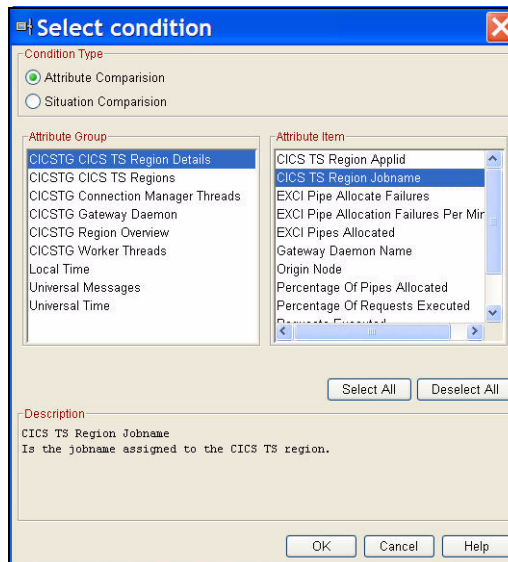


Figure 7-56 CICS TS Region Jobname condition

4. The next step is to assign values to both conditions. In our scenario, we need to reset the CICS TG health when it is 0 and its associated CICS region is available again. Therefore, we have to set the following values (Figure 7-57):
 - a. Health == 0
 - b. CICS TS Region Jobname != "

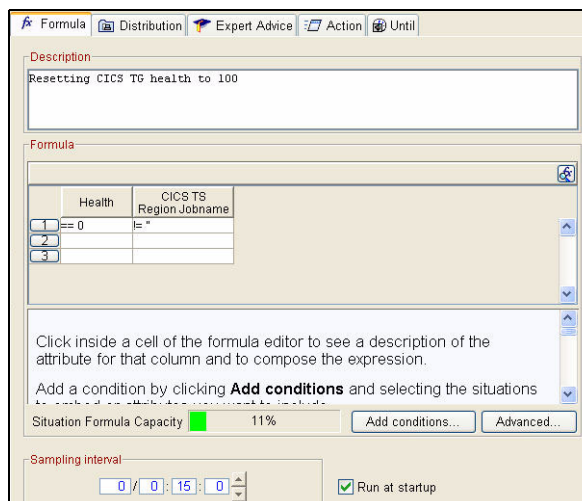


Figure 7-57 Conditions values assignment

5. We need to configure OMEGAMON XE to issue the MVS modify command when both conditions are satisfied. Click the **Action** tab and complete the form using the settings shown in Figure 7-58.

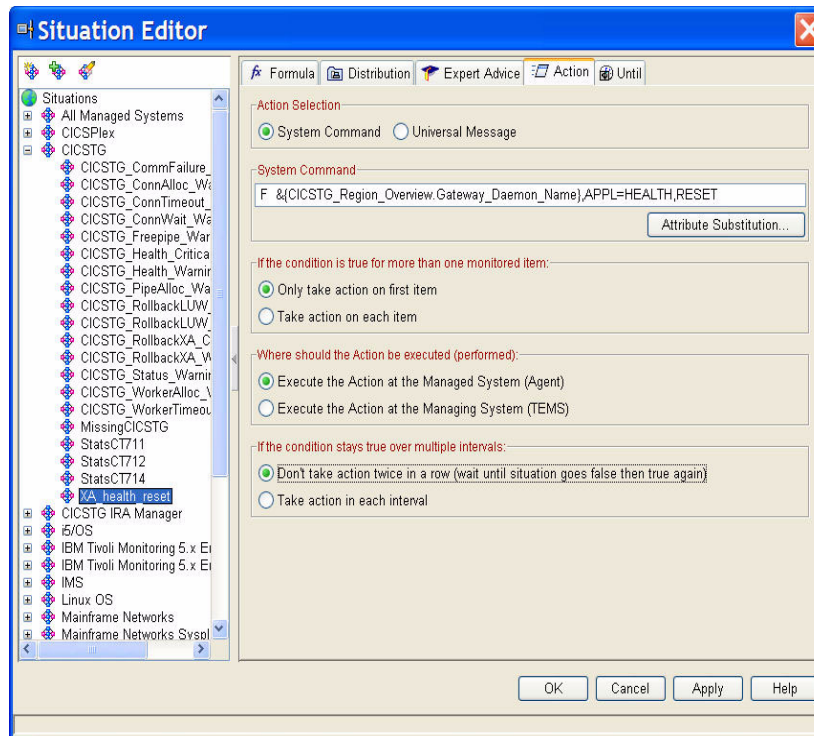


Figure 7-58 Action creation

From this point, every time the startup procedure for one of the two CICS regions has completed and the health value of its Gateway daemon is zero, OMEGAMON XE will automatically reset it to 100, informing z/OS WLM that it is again usable.

Important: When you use the CICS TG health reporting, be sure that in the `ctg.ini` file the `healthreporting` parameter is *active* (`healthreporting=on`) and the Gateway daemon has been started with this setting. If not, this will not affect the activity of OMEGAMON XE, which will continue to show the same figures we saw before, but it will prevent WLM from receiving any information about the Gateway daemon health.

Comparison of DFHXCURM versus health reporting

The two methods we have described in this section are alternatives to the problem of CICS high-availability, but do have the following significant differences:

- ▶ DFHXCURM
 - With the DFHXCURM exit method, there should be very little service interruption (that is, failing transactions) during the switch over period, but When the crashed CICS region is restarted it will not be brought back online until new EXCI pipes are allocated. This will not happen until the restart of the CICS TG, unless further EXCI pipe errors occur or pipes are deallocated because multiple CICS regions are in use (when using CTG_PIPE_REUSE=ONE).
 - The CICS TG per-server statistics will become invalid, since the re-routing happens within the EXCI and so the Gateway daemon does not know to which CICS region requests are being sent.
 - The function is reliant upon user code, which has to be carefully designed, deployed, and tested.
- ▶ Health reporting
 - With the CICS TG health reporting approach, there is likely to be a slightly longer service interruption interval (with transaction rollback), as availability has to be reported back from the Gateway daemon to WLM and on to TCP/IP.
 - Once a CICS region is restarted, it can quickly be brought back online using the CICS TG health reset command, which can be automated using OMEGAMON XE.

7.2.5 Network failure

During this test, we will aim to understand what happens to CICS TG when the network becomes unexpectedly unavailable (Figure 7-59 on page 279) and what implications this failure will have on the WebSphere and CICS transactions.

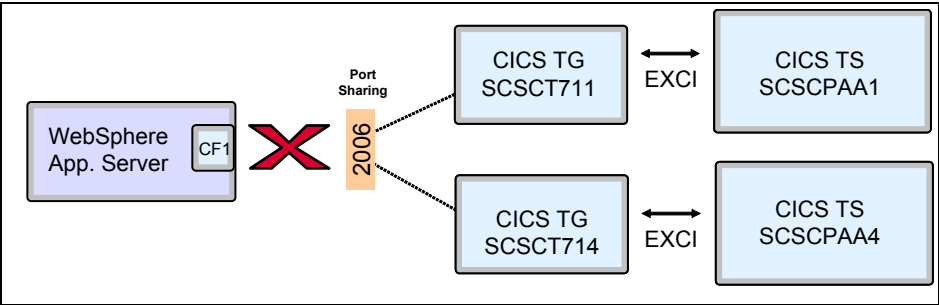


Figure 7-59 Network failure

During this scenario, we used a similar setup to the previous scenarios, but used the following settings:

- ▶ 300 concurrent requests
- ▶ 120 iterations
- ▶ Think time of four seconds
- ▶ CICS delay time of 500 ms

When we launched the workload simulation, the TEP client displayed the following information for each Gateway daemon (Figure 7-60 and Figure 7-61 on page 280).

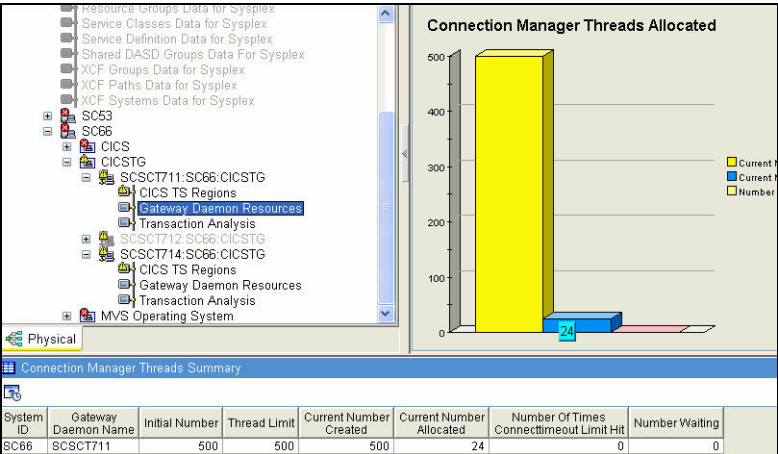


Figure 7-60 Connection Manager threads in use by CICS TG SC SCT711

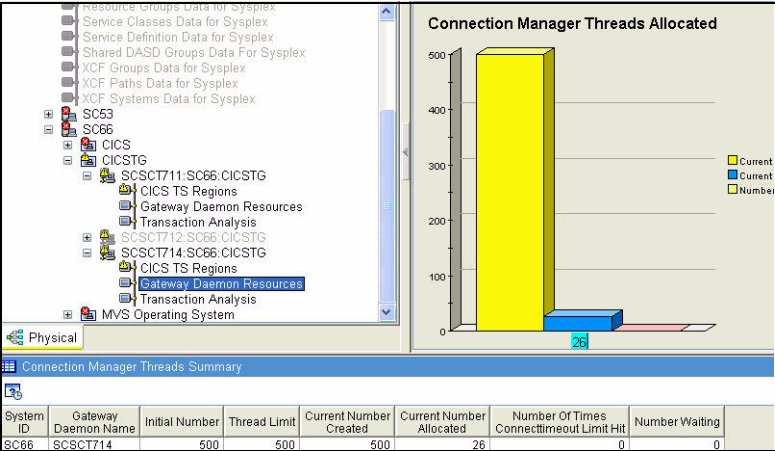


Figure 7-61 Connection Manager threads in use by CICS TG SCSCT714

These figures show that at this moment the two Gateway daemons were using 24 and 26 Connection Manager threads, respectively. Now, in order to simulate a network failure, we decided to terminate all the sockets in use by one CICS TG. However, before proceeding, we wanted to verify that the number of Connection Manager threads shown by OMEGAMON XE for the CICS TG SCSCT711 was in fact exactly the same as that returned by the NETSTAT command.

We logged on to the TSO on SC66 and ran the following MVS command where SCSCT711 is the job name of the Gateway daemon we wished to check (our output is shown in Figure 7-62 on page 281):

```
NETSTAT ALLCONN (CLIENT SCSCT711
```

MVS TCP/IP NETSTAT CS V1R8		TCPIP Name: TCPIP	09:07:58
User Id	Conn	State	
-----	----	-----	
SCSCT711	000ADFAE	Establs	
	Local Socket:	9.12.4.75..2006	
	Foreign Socket:	9.12.4.111..37022	
SCSCT711	000ADFB0	Establs	
	Local Socket:	9.12.4.75..2006	
	Foreign Socket:	9.12.4.111..37023	
SCSCT711	000ADFB1	Establs	
	Local Socket:	9.12.4.75..2006	
	Foreign Socket:	9.12.4.111..37024	
SCSCT711	000ADFB2	Establs	
	Local Socket:	9.12.4.75..2006	
	Foreign Socket:	9.12.4.111..37025	
SCSCT711	000ADFB3	Establs	
	Local Socket:	9.12.4.75..2006	
	Foreign Socket:	9.12.4.111..37026	
SCSCT711	000ADFB9	Listen	
	Local Socket:	0.0.0.0..2006	
	Foreign Socket:	0.0.0.0..0	
SCSCT711	000ADFC3	Establs	
	Local Socket:	9.12.4.75..2981	
	Foreign Socket:	9.12.4.111..37032	
SCSCT711	000ADFC4	Establs	
	Local Socket:	9.12.4.75..2006	
	Foreign Socket:	9.12.4.111..37033	
SCSCT711	000ADFC4	Establs	
	Local Socket:	9.12.4.75..2006	
	Foreign Socket:	9.12.4.111..37034	
SCSCT711	000ADFC4	Establs	
	Local Socket:	9.12.4.75..2006	
	Foreign Socket:	9.12.4.111..37035	
SCSCT711	000ADFC4	Establs	
	Local Socket:	9.12.4.75..2006	
	Foreign Socket:	9.12.4.111..37036	
SCSCT711	000ADE36	Listen	
	Local Socket:	127.0.0.1..2981	
	Foreign Socket:	0.0.0.0..0	

Figure 7-62 TSO NETSTAT command output

We counted the number of the Established connections and we found that it was exactly the same number reported by OMEGAMON XE. To kill the connections, we used the **netstat** command (with the -D flag) using a simple shell script (**killcm.sh**) to drop all the connections very quickly, exactly like a network failure does (Figure 7-63).

```
for i in $(netstat-a -E $1 | grep 2007 | grep Est | cut -f2 -d" " )
do
echo killing socket $i
netstat -D $i
done
```

Figure 7-63 Killcm.sh shell script

The output from the script is shown in Figure 7-64.

```
CICSR59 @ SC66:/u/cicsrs7>./killcm.sh SCST711
killing socket 000B825B
Connection successfully dropped
killing socket 000B825D
Connection successfully dropped
killing socket 000B8261
Connection successfully dropped
killing socket 000B826B
Connection successfully dropped
killing socket 000B8274
Connection successfully dropped
killing socket 000B8275
Connection successfully dropped
killing socket 000B8278
Connection successfully dropped
killing socket 000B8279
```

Figure 7-64 Killcm.sh shell script output

The TEP client now showed us the Connection Manager threads were immediately closed for the Gateway daemon SCST711 (Figure 7-65 on page 283).

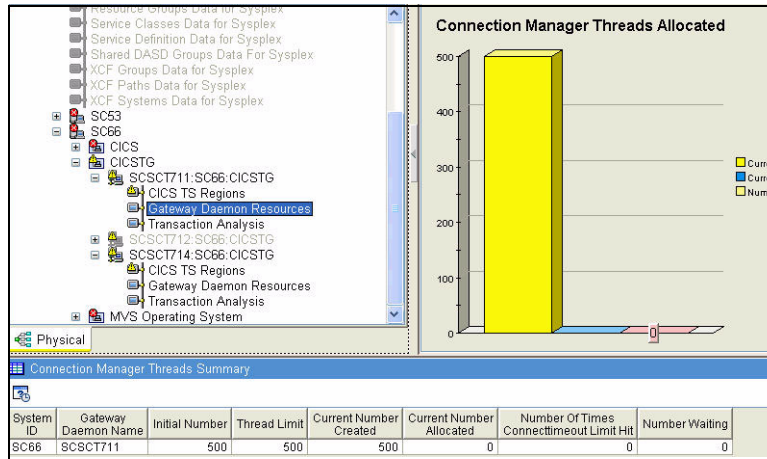


Figure 7-65 No Connection Manager threads in use by SCSCT711

However, once we refreshed the TEP client workspace, we saw that WebSphere Application Server had very quickly established new sockets and that they were already in use (Figure 7-66 and Figure 7-67 on page 284).

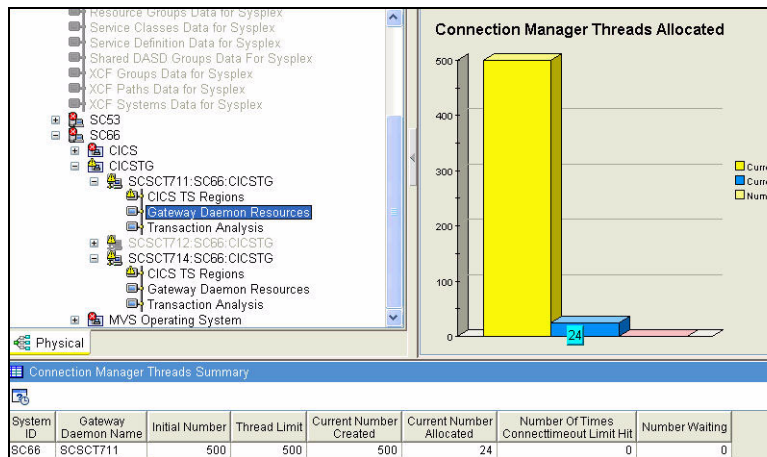


Figure 7-66 24 Connection Manager threads in use by CICS TG SCSCT711

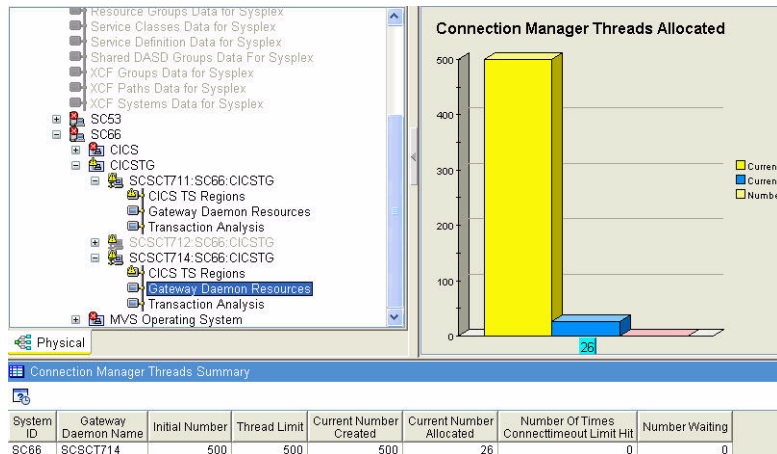


Figure 7-67 26 Connection Manager Threads in use by CICS TG SCSCT714

We repeated the test several times and we never recorded any rollbacks on the Gateway daemon that had lost the connections. Thus, in the case of network failure, our conclusion is that there should not be a significant service interruption thanks to the port sharing configuration, allowing dropped connections to be very quickly recreated by the backup Gateway daemon.

7.3 High availability - XA 2PC configuration

In this section, we will describe what we have observed and the differences we have noticed when we ran the same set of tests using two different connection factories in a true XA two-phase commit environment.

Moving our configuration from XA 1PC to XA 2PC requires the following changes:

1. At the CICS TG level, we have to switch the port sharing configuration to two stand-alone Gateway daemons, where each is independent and listening on different TCP/IP ports. In our new scenario, CICS TG SCSCT711 will listen on port 2007, and CICS TG SCSCT714 on port 2008 (Figure 7-68 on page 285).

Important: When you move from a CICS TG port sharing to a CICS TG stand-alone environment, remember to close the CICS TG master process used for the port sharing configuration, as this will no longer be necessary

2. At WebSphere level, we have to:
 - a. Use a second Connection Factory (the SCSCPAA1-XAremote1).
 - b. Customize the Custom Properties for both Connection Factories changing the Gateway daemons port numbers.
 - c. Remap the two application resource references to the JNDI names of the two Connection Factories now configured for the two stand-alone Gateway daemons. We used the following references:
 - i. eis/SCSCPAA1-XAremote
 - ii. eis/SCSCPAA1-XAremote1
 - d. Restart WebSphere Application Server.

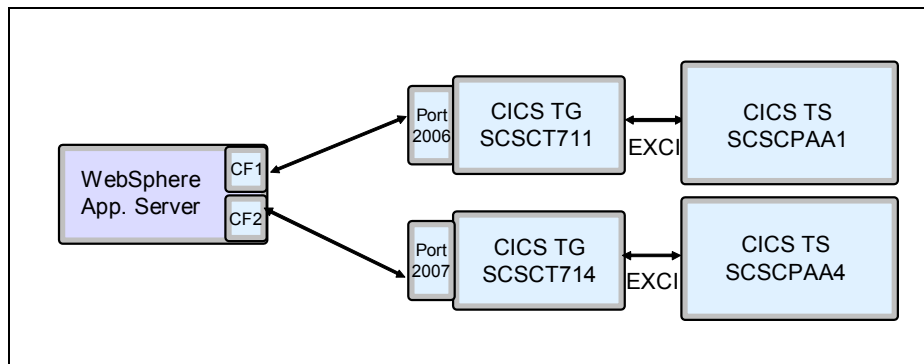


Figure 7-68 XA 2PC configuration

Note: Please refer to 4.4, “WebSphere Application Server configuration” on page 141 for details on how to modify Resource Adapters and Connection factories and how to remap Resource references for the applications.

7.3.1 Using OMEGAMON XE

In a 2PC configuration and in normal conditions, that is, when both the Gateway daemons and CICS regions are available, the WebSphere application returns the page shown in Figure 7-69. This is similar to the page returned during the 1PC tests; the distinction is that this time the first ECI request must be managed by the first Connection Factory (and so by the Gateway daemon SCST711 listening on port 2007), while the second ECI call has to be executed using the second Connection Factory (and so by the Gateway daemon SCST714 listening on port 2008).

Results	COMMAREA
Input 1:	
Output 1 using IBM037:	gaga& "e□"□'□□□□e""s"□s"TMcanaeieaeCaeeae(neeeeeeeeeee
Output 1 using default encoding:	SCSCPAA1 08/10/07 13:30:39 CICSUSER D CSMI
Output 1 in HEX:	53 43 53 43 50 41 41 31 20 30 38 2F 31 30 2F 30 37 20 31 33 3A 33 30 3A 33 39 20 20 20 20 20 20 20 00 00 00 00 00 00 00
Input 2:	
Output 2 using IBM037:	gaga& "e□"□'□□□□e""s"□s"TMcanaeieaeCaeeae(neeeeeeeeeee
Output 2 using default encoding:	SCSCPAA4 08/10/07 13:30:39 CICSUSER D CSMI
Output 2 in HEX:	53 43 53 43 50 41 41 34 20 30 38 2F 31 30 2F 30 37 20 31 33 3A 33 30 3A 33 39 20 20 20 20 20 20 20 00 00 00 00 00 00 00
Request time (ms):	560
Request succeeded.	

Figure 7-69 WebSphere Application Server output page

This also implies that the CICS region SCSCPAA1 associated with Gateway daemon SCST711 will serve the first ECI call while the second ECI request will be served by the CICS region SCSCPAA4 associated with Gateway daemon SCST714.

Because WebSphere Application Server has to manage a global transaction, it has to treat the two separate ECI requests legs as a single atomic operation. This means that if during a leg of the transaction one component of the first chain becomes unavailable, WebSphere does not have any answer back when it will ask for either a PREPARE or a COMMIT operation, so it will roll back these updates and also those belonging to the second leg (and vice versa).

So when all the components are up and running, if we run our workload, we will see zero rollbacks for the reports for the two Gateway daemons.

7.3.2 Gateway daemon unavailable

In this configuration, if for some reason (maintenance or instability) one Gateway daemon is not available, the application cannot work because there is not another Gateway daemon to replace the one that is down and to carry on with the requests that this Gateway daemon would have addressed to its default CICS.

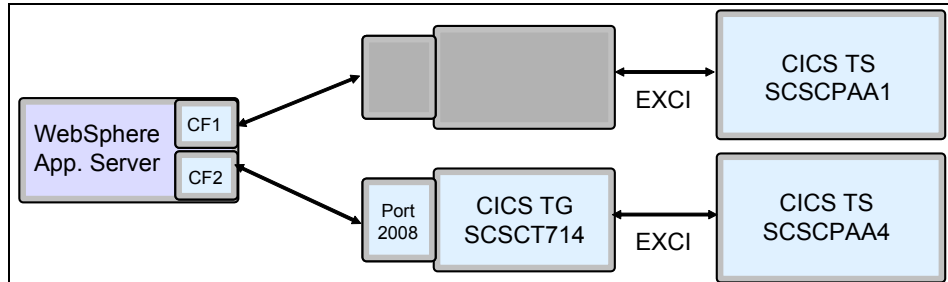


Figure 7-70 CICS TG SCST711 unavailability

If this situation occurred, WebSphere could not commit any transactions, and users would receive retryable errors until the Gateway daemon restarted. This means that in a production environment no maintenance intervention can be made without paying a service interruption. To remove this restriction, we should duplicate our environment and configure each connection factory to be served by a pair of cloned Gateway daemons.

7.3.3 Gateway daemon failure

As usual, in order to simulate a Gateway daemon crash, we canceled the job of our Gateway daemon during our HTTP workload (Figure 7-69 on page 286).

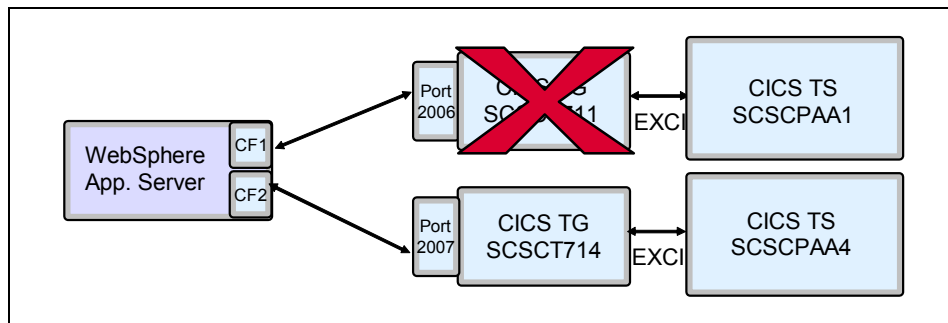


Figure 7-71 CICS TG SCST711 crash

When the Gateway daemon SCST711 crashed, we observed that:

- At the WebSphere application level, the user received the exception shown in Figure 7-72 until the Gateway daemon was restarted.

```
CICS Program name 1 (iterations): ECIPROG (1)
CICS UserID: CICSUSER
CICS mirror transaction:
WAS JNDI Name:

CICS Program name 2 (iterations): ECIPROG (1)
CICS UserID: CICSUSER
CICS mirror transaction:
WAS JNDI name:

Application trace: off
Trace: None

Warning messages:

Request
failed:
Exception occurred: 10/22/07 11:15:46.493 : javax.resource.ResourceException: enlist: caught Exception
ConnectionFactory=[com.ibm.connector2.cics.ECIConnectionFactory@cc30cc3
ConnectionManager=[ConnectionManager]@633b633b JNDI Name
SCSCPAA1-remote-XA> shareable
ManagedConnectionFactory=[com.ibm.connector2.cics.ECIManagedConnectionFactory@
e577b31c ConnectionURL="tcp://9.12.4.75" ServerName="null" PortNumber=2007"
Applid=null" ApplidQualifier=null" SocketConnectTimeout=0 Userid="Not null"
RequestExits=null" TranName=null TPName=null]]
```

Figure 7-72 WebSphere Application Server error page

- We may have some rollbacks in the Gateway daemon SCST714, depending on the moment the Gateway daemon crash happens. We know that WebSphere Application Server executes two ECI requests (the first to CICS TS SCSCPAA1 through Gateway daemon SCST711, and the second to CICS SCSCPAA4 through Gateway daemon SCST714):
 - If the crash occurs before the *prepare* phase requested by WebSphere, we will have the error exception but no rollbacks in CICS TG SCST714. This is because at the moment of failure the transaction was at a stage prior to any COMMIT or ROLLBACK decision.
 - On the contrary, if the crash happens when WebSphere has already sent the PREPARE requests to the Gateway daemons, then WebSphere Application Server has to ask the active CICS TG for the rollback of the CICS region SCST714 updates. This is because CICS region SCSCPAA1 cannot reply due to the unavailability of its CICS TG. In this situation, we will see at least one rollback.

- Until CICS TG SCST711 has restarted, no requests will be sent to the available Gateway daemon. This implies that in OMEGAMON XE workspace the value of the Number Of Requests Processed will no longer increase.

7.3.4 CICS region failure

In order to demonstrate the implications of a CICS region crash on our environment, during the workload simulation, we cancel CICS TS SCSCPAA1 using the command /c SCSCPAA1, which leads to the situation shown in Figure 7-73.

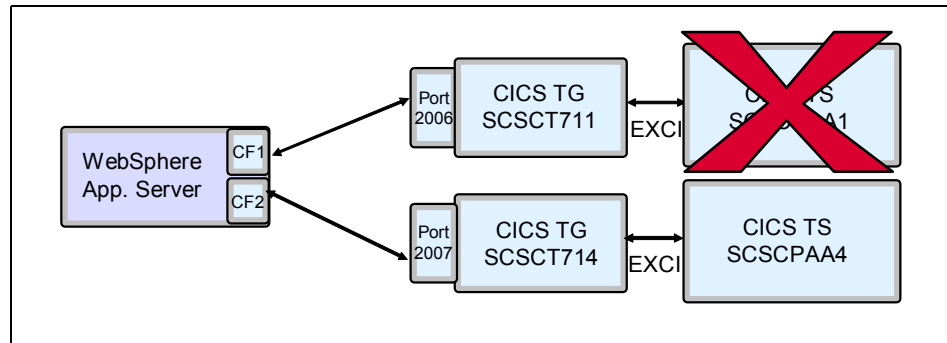


Figure 7-73 CICS TS SCSCPAA1 crash

From this test, by means of the OMEGAMON XE numbers, we could verify the following items:

- When CICS region SCSCPAA1 becomes unavailable, we see many rollbacks in its associated Gateway daemon SCST711. This number continuously increases until the Gateway daemon is restarted.
- At the same time, the transactional activity hangs for CICS TG SCST714, and the XA Transactions Committed and the Number Of Requests Processed parameters in the OMEGAMON XE Transaction Analysis perspective do not increase their values. This situation lasts until the CICS TS SCSCPAA1 restarts, but we do not see any rollback here.
- At the WebSphere Application Server level, the users receive the CTG9631E error with an error code of ECI_ERR_NO_CICS.
- After progressively decreasing, the health of SCST711 becomes zero.
- When we restarted CICS region SCSCPAA1, its Gateway daemon begins to serve successful requests and its health automatically increases up to 100. At this point, both the Gateway daemons work again as they did before the CICS region crash.

DFHXCURM exit

The DFHXCURM user exit is not applicable in such a circumstance because in a 2PC environment the CICS regions run different applications and therefore execute different transactions. We could use it only if we had a duplicated set of Gateway daemons for each connection factory.

CICS TG health reporting

When the CICS region becomes available again, the health of its Gateway daemon automatically increases its value from 0 to 100. So, in such a scenario, the usage of the CICS TG health reporting does not provide any practical benefit for the purposes of workload balancing.

7.3.5 Network failure

In our scenario, we simulated a network failure in order to understand what happens to the WebSphere and the CICS transactions when a network failure might occur in a production environment, as shown in Figure 7-74.

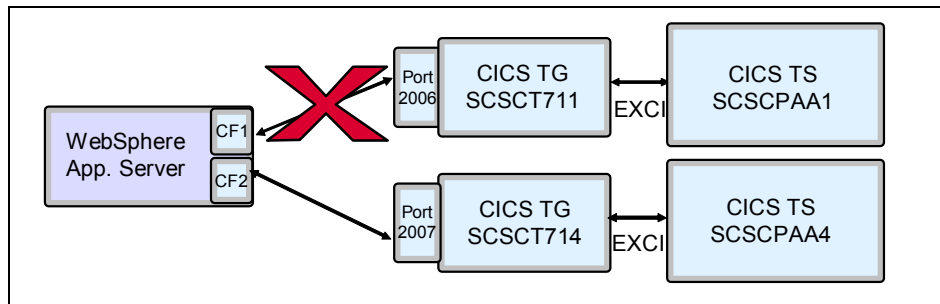


Figure 7-74 Network unavailable

If the network goes down, our expectations for this scenario are:

- ▶ To see the Gateway daemon immediately receiving new sockets/Connection Manager threads for the ones that were destroyed.
- ▶ To see some rollbacks for those transactions for which the prepare phase fails (only one or two if the system is very fast).
- ▶ To see the same rollbacks number for both the Gateway daemons, due to the fact that WebSphere Application Server has to roll back both legs of the transaction.

After checking the relationship between the **netstat output** command and the Connection Manager threads number returned by OMEGAMON XE, we executes the **killcm.sh** shell script during our usual workload simulation.

Looking at the Transaction Analysis views, we discovered the same rollback numbers for both Gateway daemons (two rollbacks each after the network interruption).

In summary, the test result was in line with our expectations and we can conclude that in the case of network failure for a 2PC environment, we have a very short service interruption and only a few rollbacks.

7.4 Summary

As we have learned from the tests results of this section, the 2PC configuration (Figure 7-75) is subject to service interruptions no matter which component becomes unavailable.

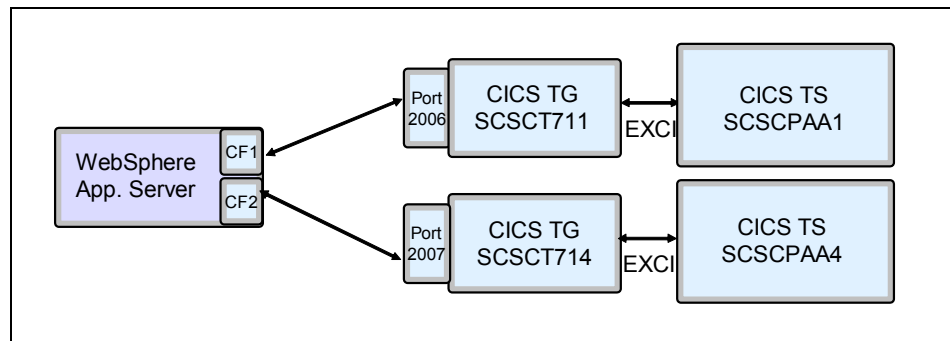


Figure 7-75 XA 2PC test environment

In order to remove this limitation and at the same time leverage the high availability benefits that the XA 1PC configuration provides, we strongly recommend duplicating the environment to provide failover at the Gateway daemon and CICS region level. Figure 7-76 shows the recommended XA 2PC scenario suitable for production environments, with redundancy at the Gateway daemon and CICS region level.

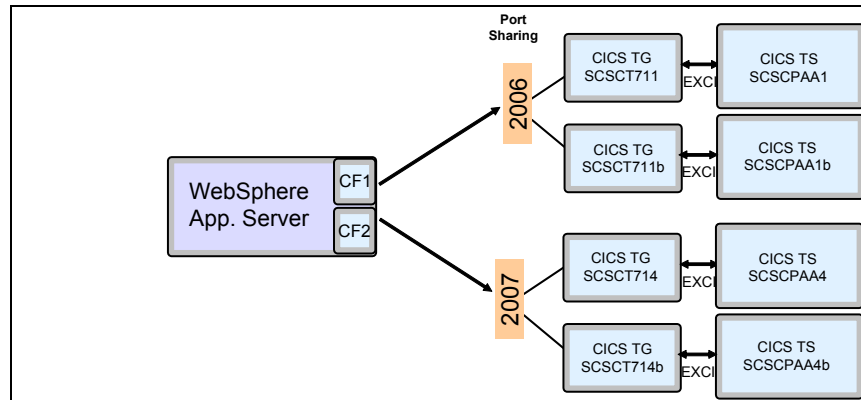


Figure 7-76 XA 2PC scenario suitable for production environments



Historical data analysis

In this chapter, we will take a look at the historical function available with OMEGAMON XE for CICS TG. We also look at the new function in CICS TG to cut System Monitoring Facility (SMF) 111 records.

8.1 Scenario introduction

This scenario will look at ways historical data can identify and help diagnose system problems. We describe the historical functions of OMEGAMON XE and take a look at the statistical data written as SMF records. Using OMEGAMON XE and SMF data, we look at one of the scenarios outlined in Chapter 6, “Diagnosing system slow downs” on page 193.

8.2 Function used

In this section, we will look at some of the functionality of OMEGAMON XE that can be used to help identify and resolve historical problems. We will also explain how to configure CICS TG to cut SMF 111 records and provide information about producing a report based on these statistics.

8.2.1 OMEGAMON XE for CICS TG historical function

Historical collection consists of near-term data storage through the Persistent Datastore (on z/OS) and the Tivoli Data Warehouse (on Windows or other distributed platforms). Therefore, you will perform configuration for historical collection in z/OS and Windows.

When historical collection is turned on, the data collected is initially stored in the Persistent Datastore (PDS) on z/OS. It will be stored there for the duration of the Warehouse Interval, which is set during Tivoli Enterprise Portal (TEP) Historical Configuration. The PDS should be set up to allow for 24x7 data collection even if no maintenance is specified. When configuring the product using the configuration tool, you can adjust the sizes of the PDS files based on your CICS TG workload and the TDW tables and collection intervals set in the TEP Historical configuration. By default, there are three datasets allocated for the PDS (&rhilev.RKGWHIS1, RKGWHIS2, and RKGWHIS3), although you can allocate more than three.

The minimum of three datasets allows continuous historical collection. The normal case is that one dataset will always be empty, one or more will be full, and one will be active. When an active dataset becomes full, the empty dataset will be activated for continued writing. When the PDS detects that there are no empty datasets left, it will find the one with the oldest data and maintain it. If the BACKUP or EXPORT options were not specified, maintenance is done within the PDS to initialize the dataset so that its status changes from full to empty. If BACKUP or EXPORT are specified, a job will be run to save the data, and then the dataset will be initialized and marked as empty. The only way that recording

would actually stop in the PDS is if the BACKUP or EXPORT was specified but the maintenance jobs fail to do their job. In this case, datasets will be taken offline until there are no more available datasets for reading or writing.

8.2.2 CICS TG SMF recording

Recording to SMF provides the capability to record interval statistics and end of day data from CICS Transaction Gateway (CICS TG) to the System Management Facility (SMF) on z/OS, following established CICS conventions.

Configuration

Recording to SMF is available by adding a parameter in the GATEWAY section of the CTG.INI file:

- ▶ statsrecording=on
- ▶ statsrecording=off

The only valid values for this parameter are on or off. Any other value will result in an error output to the JES log and the CICS TG will fail to start. The parameter and value are case insensitive. This is common to all existing Boolean parameters in the ctg.ini file.

To write to SMF, the user ID that the CICS Gateway daemon runs under must be permitted READ access to the BPX.SMF facility. An example of the syntax is shown here:

```
PERMIT BPX.SMF CLASS(FACILITY) ACCESS(READ) ID(USERID)
```

Configuring this permission is a mandatory step for the installation and the upgrade processes. The RACF and UNIX System Services documentation gives further details on this topic.

Interval correlation number

Interval statistics are captured at predefined intervals by issuing requests for them to be recorded at the correct time. These two features will interact in the following way:

- ▶ The interval statistics component will periodically request the SMF recording facility to record statistics information.
- ▶ The interval statistics component will take a snapshot of all current statistics and send them to the SMF recorder.
- ▶ The SMF recorder will format the statistics into the documented SMF record format. This data will then be output to the SMF facility. Every time the interval statistics component issues a request to record statistics, a unique number

will be generated. This number is known as the *interval correlation number* and is unique for a specific CICS TG instance over the lifetime of that CICS TG.

- The interval correlation number is included as an external field in every SMF record header.

The SMF subsystem allows a maximum of 30 KB per record. If the formatted data for a statistics request exceeds 30 KB, the data is split across multiple SMF records. Each SMF record is given the same interval correlation number. This correlation allows the records relevant to an individual statistics interval to be linked together.

Accessing the CICS TG SMF records

To enable you to access the CICS TG SMF records, you need to enable SMF recording in the CICS TG as part of your CICS TG implementation (see “Activating SMF recording” on page 51).

Once this is enabled, you need to compile and link-edit the SMF record viewer sample program called CTGSMFRD (see “JCL to link-edit CTGSMFRD” on page 372).

Once compiled and link-edited, you then need to run a job to import the SMF records into a dataset, and then you can extract them and run them against the CTGSMFRD program. See “JCL to extract type111 records from SMF” on page 374 for the sample job we ran.

Note: You must link-edit the CTGSMFRD into a PDSE library.

We are now in a position where we have the SMF data in dataset SMFDATA.CTGRECS.REDBOOKS, so the next step is to run the Sample JCL to run the SMF record viewer sample program CTGSMFRD. See “JCL to run SORT SMF records” on page 375 for an example of the job we ran.

Upon viewing the output, we noted that the records were being reported in a random order. We wanted to have the report sorted by Gateway daemon and then time sequence. To achieve this setup, we ran a sort (DFSORT™) as a separate step between the extraction and the CTGSMFRD program step (see “JCL to run SORT SMF records” on page 375). The sort ordered the statistics by applid, then day, and then time, using the fields SMF111_SMFSPN, CTG_COLDATE, and CTG_COLTIME. The details of these fields are shown in Table 8-1 on page 297.

Table 8-1 Fields used in SMF sort step

Field description	Type	Offset	Length	Description
SMF111_SMFS PN	Char	10	8	Gateway daemon identifier CICS TG APPLID
CTG_COLDATE	Binary	24	4	Collection date (OCYYDDD+) Local time
CTG_COLTIME	Binary	28	4	Collection time (OOHHMMSS) Local time

The format of the SMF records can be found in the section SMF product section data structure in the CICS TG InfoCenter.

Note: The above field offsets show their relative position within the product section of the SMF records. The product section follows a SMF header of 44 bytes in length. To calculate the offset within the SMF record, we added 44 to these offsets. DFSORT offsets start at position 1, not 0, so for the sort parms of DFSORT, we added an additional 1 to the start position.

8.3 Using historical data to diagnose system problems

This scenario will look at ways historical data can identify and help diagnose system problems. We describe the problem setup, the warning signs that indicate that a problem has occurred, the diagnosis steps, and the actions we took as a result. We will run a long duration (12 hour) workload to examine the historical function of OMEGAMON XE and records written to SMF as type 111.

8.3.1 Creating the base workload

For this test, we used a single Gateway daemon (SCSCT711) with one CICS region (SCSCPAA1) to simulate an overnight workload where one of the Gateway daemons in our base configuration has been stopped. Figure 8-1 shows the basic configuration for the test; Gateway daemon SCSC714 and CICS region SCSCPAA4 have been shut down and are shown as faded out.

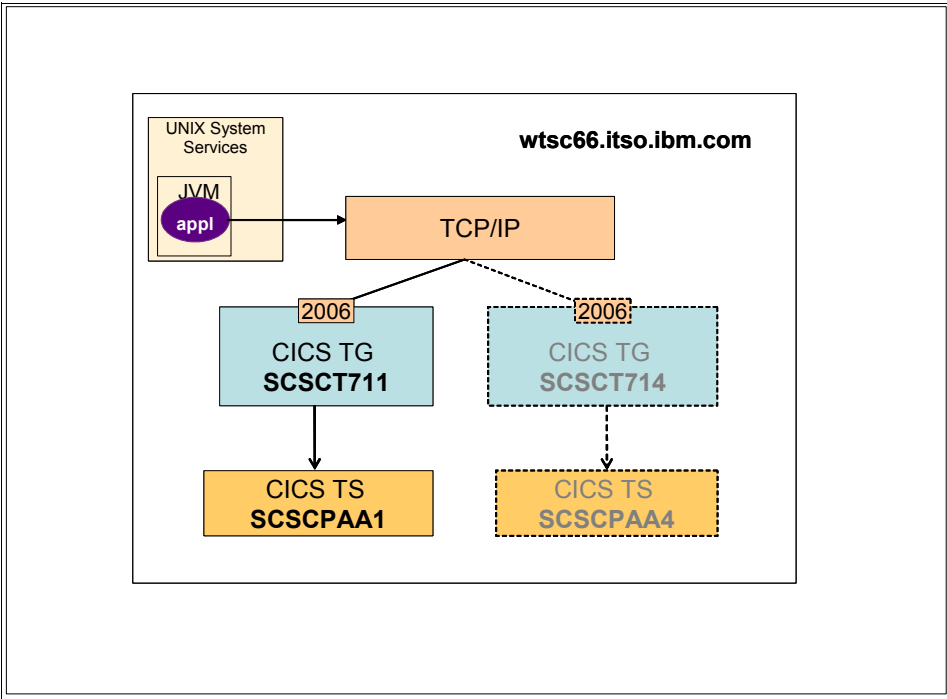


Figure 8-1 Basic configuration with only one Gateway daemon and CICS region active

The key elements of the CICS TG configuration file are shown in Table 8-2.

Table 8-2 Key setup configuration parameters

Key parameter	Setting
Maximum Connection Managerthreads	500
Maximum worker threads	100
Workertimeout	1000 ms
Statistic interval	010000 (60 minutes)
Statistic end-of-day	000000 (midnight)

Using the same rule of thumb equations as outlined in 6.3.2, “Gateway daemon Worker thread constraint” on page 213, we calculated the workload driver throughput using the formula shown here:

workload driver $TPS = numUsers / (thinkTm + CICS RespTime)$

Where TPS = transactions per second, CICS RespTime = average response time through the CICS region, numUsers = number of concurrent users, and thinkTm = average user think time.

The above formula is intended as a very rough general rule. All systems have different dynamics, so this formula should be used as a starting point for tuning the system only. All our timings are in seconds.

We set the system up as shown in Table 8-2 on page 298. The workload driver was started using the application settings shown in Table 8-3.

Table 8-3 Application settings for the base workload

Key parameter	Setting
Number of concurrent connections from the workload driver (users)	300
Think time between requests on each connection on the workload driver.	1000 ms
Approximate CICS transaction time (delay in CICS program)	500 ms

Based on these settings and using our formula, we predicted that the best achievable CICS throughput was the number of Worker threads/CICS resp or $100/0.5 = 200$ TPS, and the maximum workload driver rate was the number of users/(thinktime + RespTime) or $300 / (1 + 0.5) = 200$ TPS. Therefore, in theory, our Gateway daemon should cope with the application workload.

We then ran the base workload starting at 18.15 local time.

Note: The time zone of the machine running the test was EST5EDT, which was five hours behind the users. The workload was started at 13:13 EST and 18:13 local time.

After the base workload had been running for five minutes, we issued a series of modify commands at five minute intervals, through SDSF, against the Gateway daemon started task. These commands were issued to capture and display the statistics of the Gateway daemon. Information from these samples would be used to assess whether the Gateway daemon was actually capable of handling the workload. The syntax of the MVS modify command is /F SCSC711,APPL=STATS,GS.

Some of the key statistics returned are shown in Table 8-4.

Note: The samples occurred within the same stats interval that was reset at 13:00. Therefore, figures for GD_IALLREQ would be cumulative totals since the reset time.

Table 8-4 An extract from the statistics gathered using a modify command

Time	CM_CALLOC	WT_CALLOC	WT_LTIMEOUTS	GD_IAVRESP	GD_IALLREQ	CSSCSCPAA1_IAVRESP
13.16	300	100	0	542	14097	502
13.21	300	100	0	535	72992	501
13.26	300	94	0	534	131840	501

Based on the figures shown in Table 8-4, we calculated the actual TPS of the Gateway daemon and application driver:

- ▶ CICS throughput $100 / 0.501 = 199$ TPS
- ▶ Application driver $= 300 / (1.0 + 0.53) = 196$ TPS

As expected, the Gateway daemon was coping with the base workload, but we are running at the limit of the Gateway daemon's capacity. There would be little margin for an increase in this workload before requests start to back up in the Gateway daemon.

GD_IALLREQ represents a cumulative total of requests through the Gateway daemon that these stats are sampling. If we take these values and calculate the delta between the samples and convert that to a TPS figure, we get 196.16 and 196.31 for samples 2 and 3, respectively. These TPS figures confirm the Gateway is operating near full capacity, as predicted in our TPS calculation.

Finally, CICS region response times (CSSCSCPAA1_IAVRESP) show that CICS was handling the workload satisfactorily, consistently returning an average response times slightly higher than the 500 ms delay.

We concluded that the configuration was capable of handling the workload and we would continue with the test. The workload was then left to run for 12 hours, effectively overnight.

8.3.2 Gathering historical data from SMF

The Gateway daemon was configured to capture statistics every 60 minutes (statint=010000 in the configuration file ctg.ini) on the hour. These records are written to one of the z/OS SMF datasets as SMF type111 records. Example 8-1 shows that the MVS command D SMF was used to display the status of the SMF datasets. When full, the SMF datasets switch and an archive job will copy the CICS records to a Generation Dataset Group (GDG), prefixed with SMFDATA.CICSRECS and suffixed with a generation number, for example, SMFDATA.CICSRECS.G4906V00. More information about SMF recording can be found in 8.2.2, “CICS TG SMF recording” on page 295.

Example 8-1 MVS command used to display the SMF dataset and its status

D SMF					
RESPONSE=SC66					
IEE974I 04.15.28 SMF DATA SETS 426					
NAME		VOLSER	SIZE(BLKS)	%FULL	STATUS
P-SYS1.SC66.MAN1		PAG660	3000	56	ACTIVE
S-SYS1.SC66.MAN2		PAG660	3000	0	ALTERNATE
S-SYS1.SC66.MAN3		PAG660	3000	0	ALTERNATE

8.3.3 Gathering historical data using OMEGAMON XE history data

History collection was turned on in OMEGAMON XE, causing historical statistics to be written to the Persistent Datastore (see 3.3.4, “Historical configuration” on page 112). The frequency of these writes and the location of the data stored is based on history collection configuration settings. To examine and change the history collection configuration settings, use the Tivoli Enterprise Portal (TEP) client as follows:

1. Click the History Configuration icon on the tool bar, as shown in Figure 8-2.

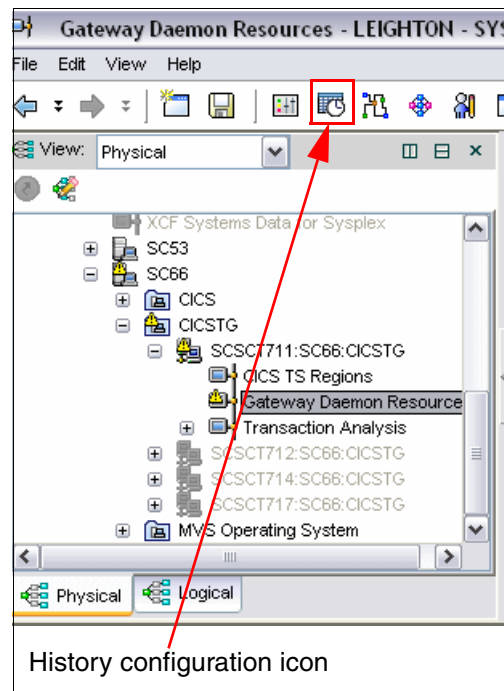


Figure 8-2 Selecting the History configuration window

2. Select the Select a Product box and select **OMEGAMON XE for CICS TG on z/OS**. The history collection configuration window appears, as shown in Figure 8-3 on page 303.

History Collection Configuration

Select a product
OMEGAMON XE for CICS TG on z/OS V4.1.0

Select Attribute Groups

Group	Collection	Collection Interval	Collection Location	Warehouse Interval	Summarize Yearly	Prune Yearly	Summarize Quarterly
CICSTG_Connection_Manager_Threads	Started (3)	5 minutes	TEMA	1 hour			
CICSTG_CICS_TS_Region_Details	Started (3)	5 minutes	TEMA	1 hour			
CICSTG_CICS_TS_Regions	Started (3)	5 minutes	TEMA	1 hour			
CICSTG_Gateway_Daemon	Started (3)	5 minutes	TEMA	1 hour			
CICSTG_Region_Overview	Started (3)	5 minutes	TEMA	1 hour			
CICSTG_Worker_Threads	Started (3)	5 minutes	TEMA	1 hour			

Configuration Controls

Collection Interval: 5 minutes

Collection Location: TEMA

Warehouse Interval: 1 hour

Summarization:

- ☐ Yearly
- ☐ Quarterly
- ☐ Monthly
- ☐ Weekly
- ☐ Daily
- ☒ Hourly

Pruning:

- ☐ Yearly keep [] Years
- ☐ Quarterly keep [] Years
- ☐ Monthly keep [] Months
- ☐ Weekly keep [] Months
- ☐ Daily keep [] Days
- ☐ Hourly keep [] Days
- ☐ Detailed data keep [] Days

Configure Groups Unconfigure Groups Show Default Groups **Start Collection** Stop Collection Refresh Status

Close Help

Figure 8-3 History Collection Configuration window showing collection group information

We set the history collection configuration settings to:

Collection Interval The amount of time between data collection. We set this to five minutes for each group.

Collection Location The location from where the data will be collected. We set this to the Tivoli Enterprise Monitoring Agent (TEMA). See 3.3.4, “Historical configuration” on page 112 for an explanation about the collection location.

Warehouse Interval The interval of time for which you want data sent to the warehouse. This interval was set to one hour.

To change the history configuration settings, click the row you wish to change and the Configuration Controls menu will change to show the current settings for that attribute group (see Figure 8-3). Use the drop-down box to select a predefined value.

To implement the change, click the **Configure Groups** button and the Collection row for the group should change to Started. If the Collection row for the group does not show Started, click the **Start Collection** button.

For further information about configuring the collection of historical stats, see the section Configuring historical data in the OMEGAMON XE for CICS TG InfoCenter.

8.3.4 Spotting the warning signs

The morning after the workload finished (a duration of 12 hours), there were reports of an increase in transaction response time during the night and complaints of application timeouts. However, investigations into these incidents did not occur during the night and diagnosis had to wait until after the test had finished. This took place the morning following the overnight run.

Using the TEP client (see Figure 8-4 on page 305), we were able to examine Gateway daemon resources as they were after the workload had finished. Most totals were zeroes because they had been reset when the stats interval expired (every hour). However, lifetime stats are cumulative totals that do not get reset. One such total Number Of Times Workertimeout Limit Hit was high at 11328. This gave us a clue that there was a possible problem with Worker threads. We would have to use historical data to investigate this further.

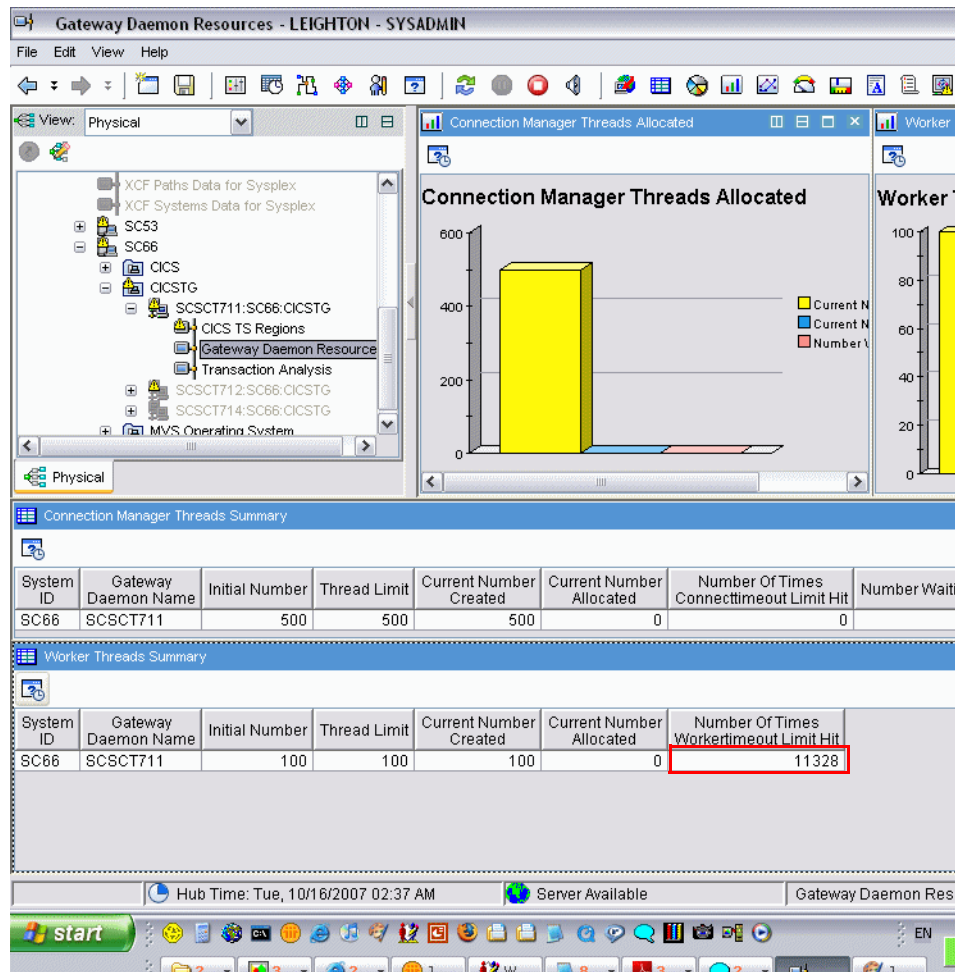


Figure 8-4 TEP Gateway daemon resources workspace

8.3.5 Diagnosis using SMF statistics

We wanted to look at how historic data written to SMF could be used to identify and diagnose system problems. To do this task, we ran a batch job (see “JCL to extract type111 records from SMF” on page 374 for more details) using archived SMF GDGs containing the interval statistics captured overnight as input to extract the type111 records to an output dataset SMFDATA.CTGRECS.REDBOOKS. This dataset was then used as input to the SMF record viewer sample program CTGSMFRD (see “JCL to run SORT SMF records” on page 375 for more details). The program CTGSMFRD produced a

report containing 12 interval records; the first interval is shown in Appendix B, “SMF Historical Data” on page 365.

Note: The time zone of the machine running the test was EST, which was five hours behind the users. The workload was started at 13:15 EST and 18:15 local time.

Table 8-5 shows a summary of the key values extracted from the report produced by CTGSMFRD.

Table 8-5 An extract taken from the CTGSMFRD report

Interval/ time	CM_ CALLOC	CM_ CWAITING	WT_ CALLOC	WT_ LTIMEOUTS	GD_ IAVRESP	GD_IALL REQ	CTG_CSX_I AVRESP
1/14.00	400	146	97	3406	991	518298	513
2/15.00	400	152	100	10309	1174	696945	515
3/16.00	300	0	33	11328	1044	554304	503
4/17.00	0	0	0	11328	994	179247	500
5/18.00	0	0	0	11328	994	0	0
6/19.00	0	0	0	11328	994	0	0
Values repeated as above							
12/01.00	0	0	0	11328	994	0	0

Here are some observations about the SMF statistics:

Stats intervals In 5 - 12, the number of requests GD_IALLREQ has dropped to zero. The workload appears to have stopped some time between 17.00 and 18.00.

The workload was started at 13:15. Therefore, interval 1 is partial, representing only 45 minutes of the hour. It also shows that the number of allocated connection managers (CM_CALLOC) has increased to 400 from the original value of 300.

Interval 3 shows the number of allocated connections (CM_CALLOC) has returned to 300.

Interval 2 is a full hour of the higher workload, so the number of allocated connections is at 400.

Gateway daemon response time

This has increased from an average of around 530 ms at the start of the test to over 900 ms. User response times have increased by approximately 70%.

CICS response time (CTG_CSX_IHVRESP)

This is similar to the baseline values shown in 8.3.1, “Creating the base workload” on page 298. Also, the number of requests (GD_IALLREQ) represents a TPS of 914 going through the Gateway daemon, similar to the earlier baseline values. Both of these figures prove the CICS region is still operating at near full capacity during these intervals.

CM_CALLOC

An increase in the current number of allocated connection managers (CM_CALLOC) to 400 indicates a change in workload.

CM_CWAITING

The size and frequency of the new workload is unknown. However, a clue to this extra workload is given in the CM_CWAITING column, where the values of 146 and 152 represent a significant amount of requests queued in the Gateway daemon waiting for a Worker thread. The extra workload would appear to be causing requests to back up in the Gateway daemon waiting for a Worker thread.

WT_LTIMEOUTS

Requests that wait longer than the workertimeout value specified in the configuration file ctg.ini will time out after 1000 ms. WT_LTIMEOUTS shows that during the test, 11328 requests timed out. However, we do not know whether this was a isolated incident or a gradual trend during the interval.

In conclusion, the SMF statistics show that an increase in the workload has resulted in requests waiting in the Gateway daemon and in some cases these requests end with timeouts. However, CICS response times and TPS have not dropped. Therefore, the problem does not appear to be a CICS related issue.

The SMF statistics taken at one hour intervals can only give a high level overview of the problem. We need more detailed data, at a more regular interval, to perform any further diagnosis. The `statint` value in the Gateway daemon configuration file could be changed to a smaller interval to capture more statistics. This would mean recycling the Gateway daemon and waiting for a re-occurrence of the problem. We could also look at what OMEGAMON XE has to offer in terms of historical data.

8.3.6 Diagnosis using OMEGAMON XE

To investigate the problem using OMEGAMON XE and its historical statistic collection function, we used the Tivoli Enterprise Portal (TEP) client. To examine Gateway daemon resources statistics for Gateway daemon SCSC711, we opened nodes and drilled down to the Gateway daemon Resources workspace in the navigation view (see Figure 8-5).

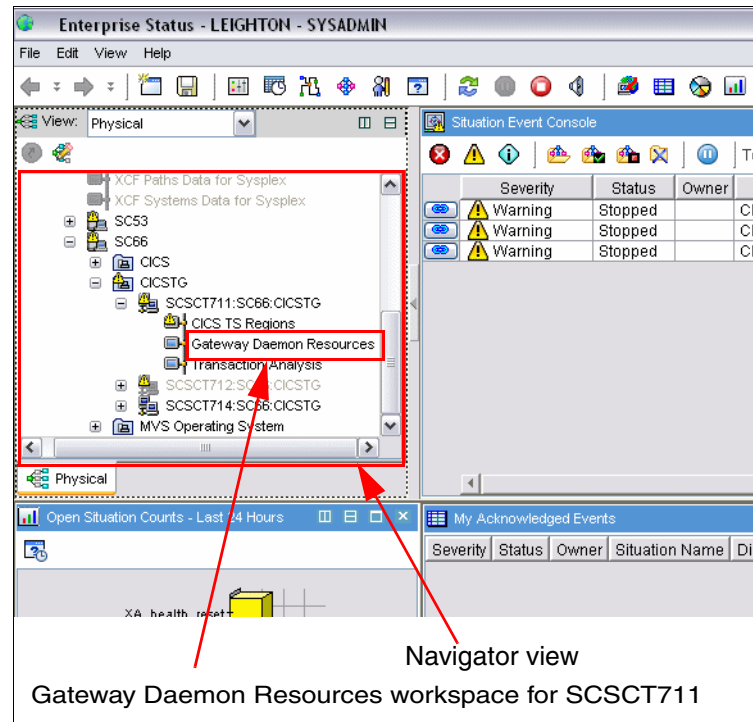


Figure 8-5 Selecting the Gateway daemon Resources window using the navigator view

The resultant window shows the *real time* stats for Gateway daemon resources (see Figure 8-6 on page 309) as single rows within a table in the Connection Manager Threads summary and Worker threads summary views.

Tip: Pressing PF5 will refresh the data in the views.

We want to examine stats collected in the Persistent Datastore for the previous 24 hours. We selected the history time span icon on the Worker thread summary view (see Figure 8-6 on page 309).

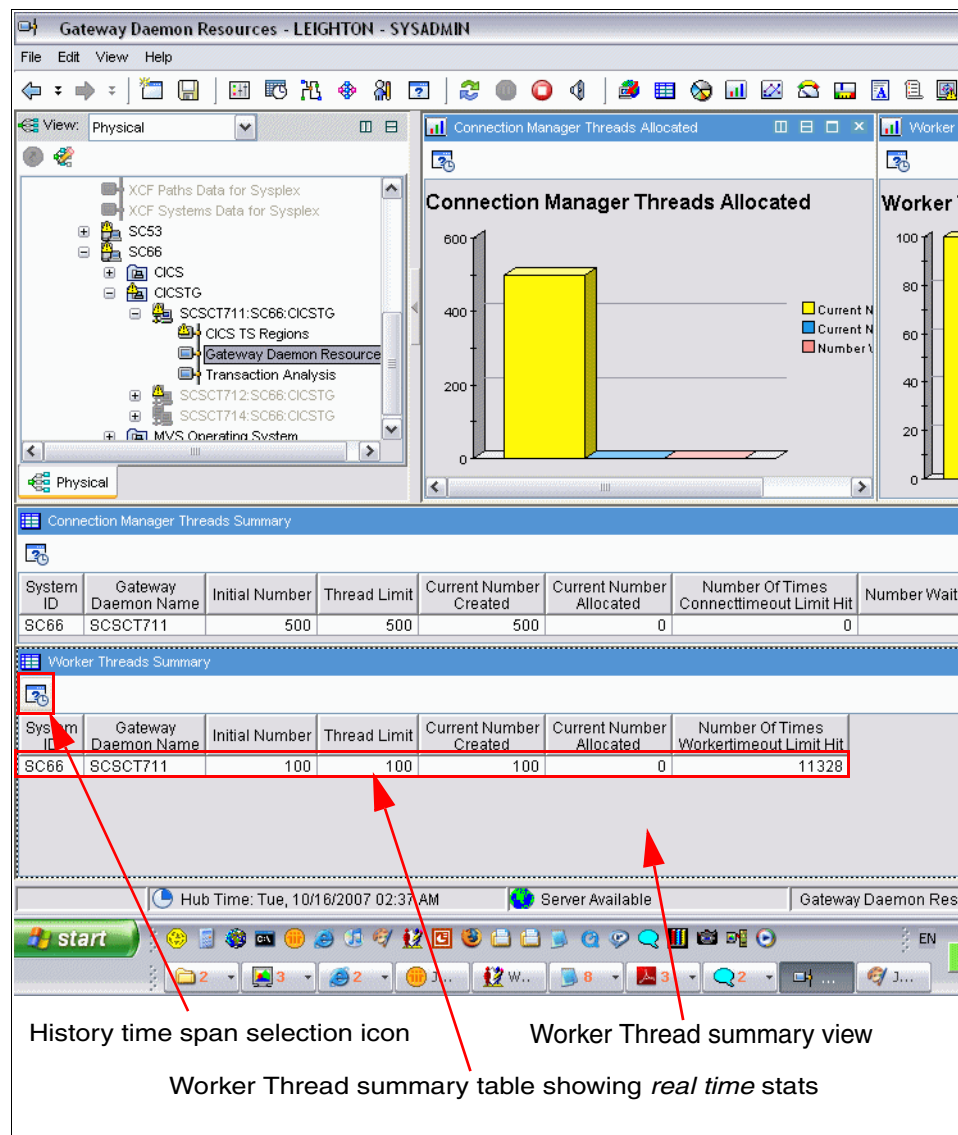


Figure 8-6 Gateway daemon resource view showing real time data

The Select the time span view appeared, as shown in Figure 8-7. We clicked the **Last** button and entered “24” and “hours” in the boxes and clicked **OK**. This caused OMEGAMON XE to retrieve the last 24 hours of Worker thread summary stats.

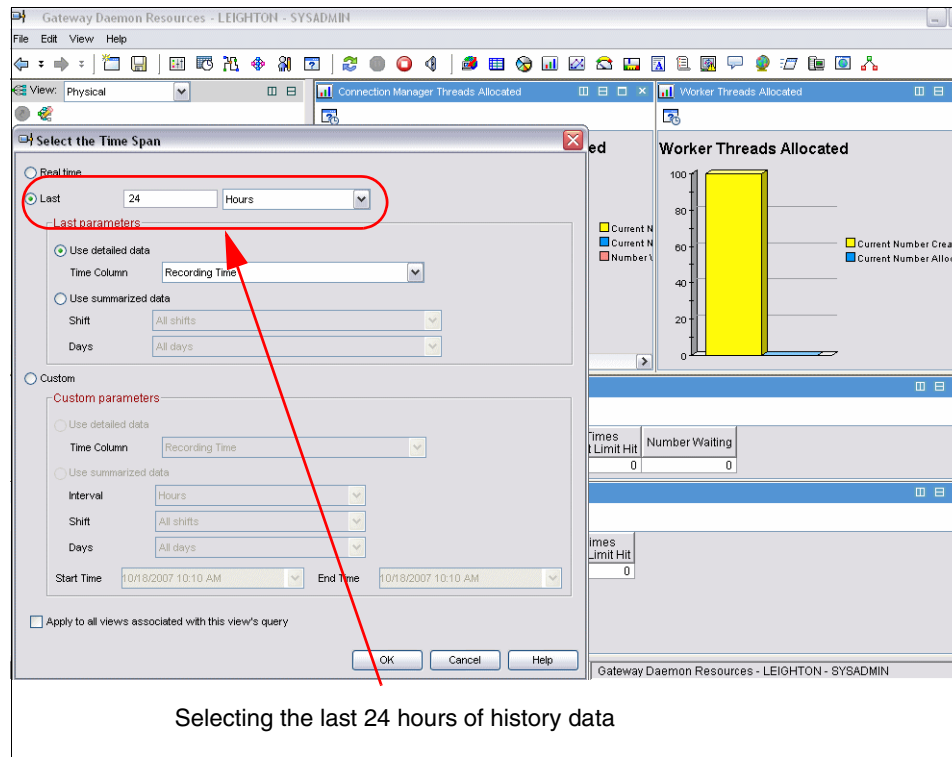


Figure 8-7 Select the time span view

The Worker threads summary table changed, as shown in Figure 8-8. The table expanded to contain rows of statistic data for the previous 24 hours. These roes are presented in pages of information, and the page number is shown in the top right of the view. The scroll bar can be used to move up and down the rows and between pages.

Tip: The double arrow scroll button moves between pages.

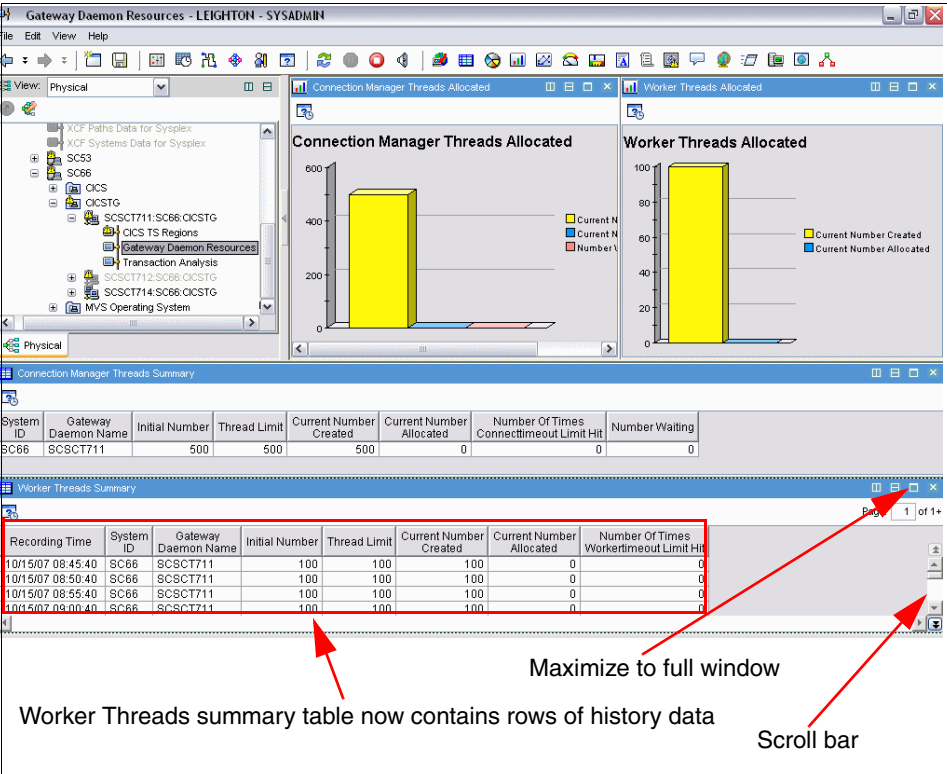
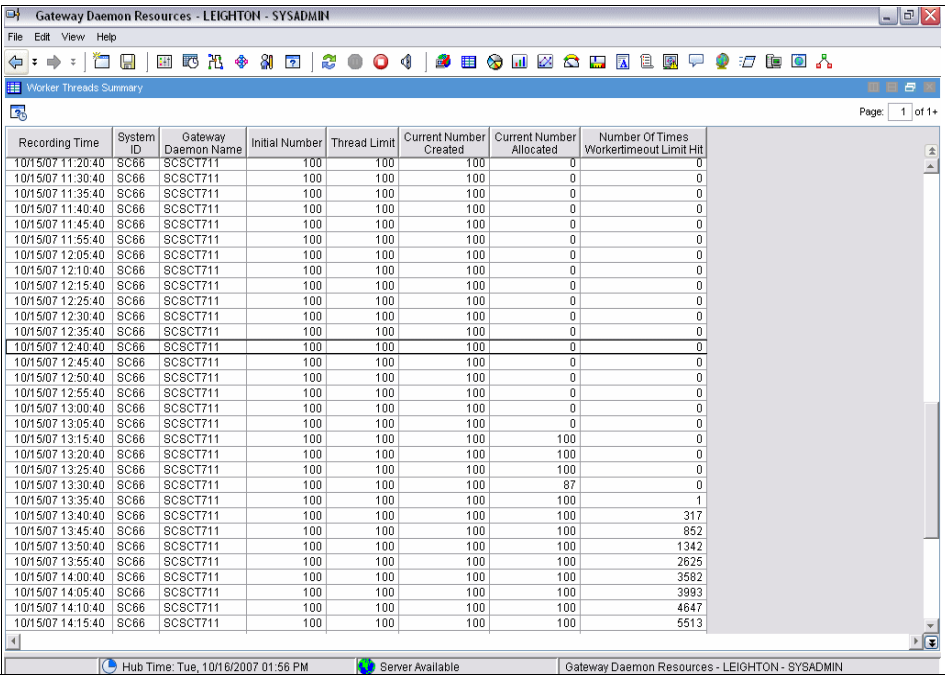


Figure 8-8 Worker threads summary table containing history data

It can be difficult to get a good overview of the data using this restricted view. An expanded view of the data is useful and can be obtained by clicking the **Maximize** button. The presents a full window of statistical data for this table, as shown in Figure 8-9.



The screenshot shows a window titled "Gateway Daemon Resources - LEIGHTON - SYSADMIN". The window contains a table titled "Worker Threads Summary". The table has the following columns: Recording Time, System ID, Gateway Daemon Name, Initial Number, Thread Limit, Current Number Created, Current Number Allocated, and Number Of Times Workertimeout Limit Hit. The table displays data for various recording times from 10/15/07 11:20:40 to 10/15/07 14:15:40. The data shows that the Initial Number and Thread Limit are consistently 100. The Current Number Created and Current Number Allocated are consistently 100. The Number Of Times Workertimeout Limit Hit varies, starting at 0 and increasing to 5513 by 10/15/07 14:15:40.

Recording Time	System ID	Gateway Daemon Name	Initial Number	Thread Limit	Current Number Created	Current Number Allocated	Number Of Times Workertimeout Limit Hit
10/15/07 11:20:40	SC66	SCSCT711	100	100	100	0	0
10/15/07 11:30:40	SC66	SCSCT711	100	100	100	0	0
10/15/07 11:35:40	SC66	SCSCT711	100	100	100	0	0
10/15/07 11:40:40	SC66	SCSCT711	100	100	100	0	0
10/15/07 11:45:40	SC66	SCSCT711	100	100	100	0	0
10/15/07 11:55:40	SC66	SCSCT711	100	100	100	0	0
10/15/07 12:05:40	SC66	SCSCT711	100	100	100	0	0
10/15/07 12:10:40	SC66	SCSCT711	100	100	100	0	0
10/15/07 12:15:40	SC66	SCSCT711	100	100	100	0	0
10/15/07 12:25:40	SC66	SCSCT711	100	100	100	0	0
10/15/07 12:30:40	SC66	SCSCT711	100	100	100	0	0
10/15/07 12:35:40	SC66	SCSCT711	100	100	100	0	0
10/15/07 12:40:40	SC66	SCSCT711	100	100	100	0	0
10/15/07 12:45:40	SC66	SCSCT711	100	100	100	0	0
10/15/07 12:50:40	SC66	SCSCT711	100	100	100	0	0
10/15/07 12:55:40	SC66	SCSCT711	100	100	100	0	0
10/15/07 13:00:40	SC66	SCSCT711	100	100	100	0	0
10/15/07 13:05:40	SC66	SCSCT711	100	100	100	0	0
10/15/07 13:15:40	SC66	SCSCT711	100	100	100	100	0
10/15/07 13:20:40	SC66	SCSCT711	100	100	100	100	0
10/15/07 13:25:40	SC66	SCSCT711	100	100	100	100	0
10/15/07 13:30:40	SC66	SCSCT711	100	100	100	87	0
10/15/07 13:35:40	SC66	SCSCT711	100	100	100	100	1
10/15/07 13:40:40	SC66	SCSCT711	100	100	100	100	317
10/15/07 13:45:40	SC66	SCSCT711	100	100	100	100	852
10/15/07 13:50:40	SC66	SCSCT711	100	100	100	100	1342
10/15/07 13:55:40	SC66	SCSCT711	100	100	100	100	2625
10/15/07 14:00:40	SC66	SCSCT711	100	100	100	100	3582
10/15/07 14:05:40	SC66	SCSCT711	100	100	100	100	3993
10/15/07 14:10:40	SC66	SCSCT711	100	100	100	100	4647
10/15/07 14:15:40	SC66	SCSCT711	100	100	100	100	5513

Figure 8-9 Expanded Worker threads summary information

Using the scroll bar to move down the rows, it was possible to locate the start time of our workload.

We could now see more detailed information about the problem. But we wanted to concentrate on a specific time period within this table. For example, we were not interested in stats from the time prior to the start of the test. To filter out rows from the table, we went back to the Select the time span view, shown in Figure 8-7 on page 310, clicked the **Custom** radio button, then specified a start and end time using the pop-up menu, as shown in Figure 8-10 on page 313.

Tip: Enter the end and start time required using the hours, minutes, and seconds boxes. To select this time, you have to click the highlighted **Day** button in the calendar. There is no **OK** button; if you click outside the pop-up menu, your selection will be lost.

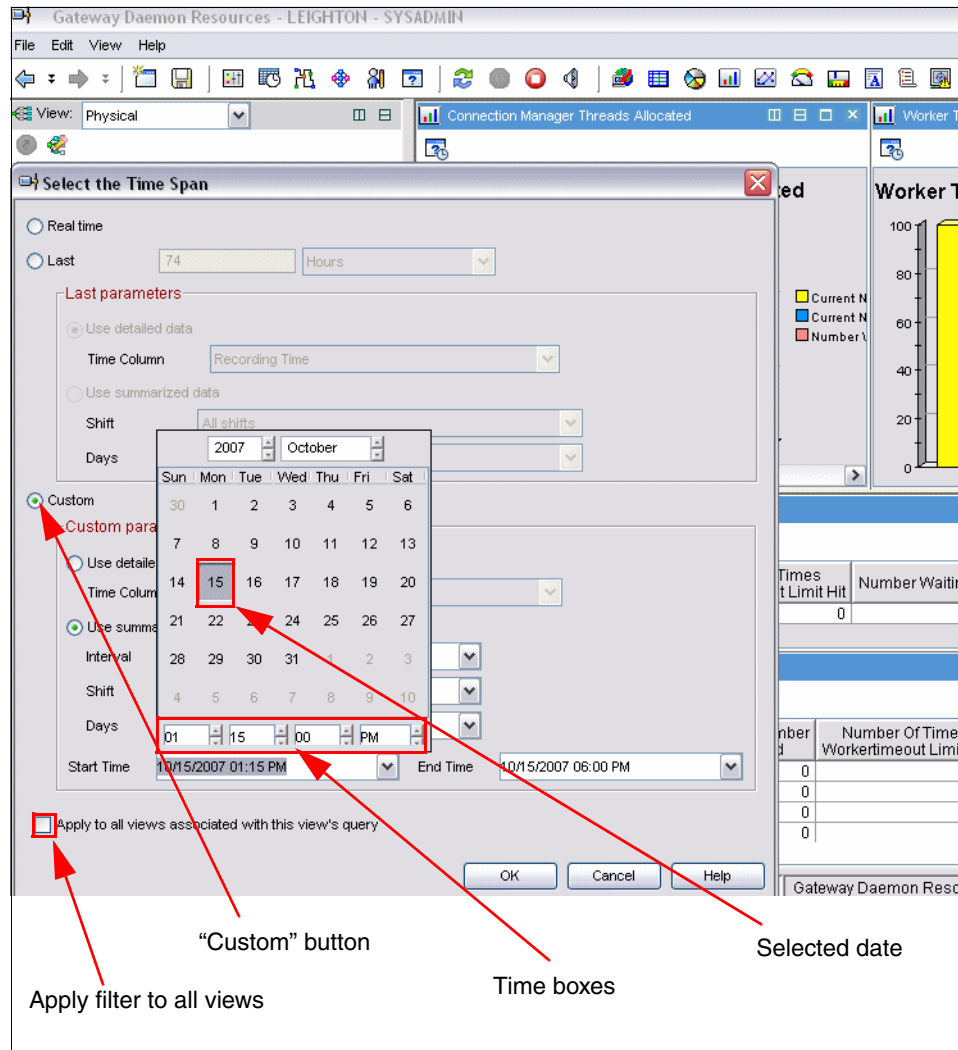


Figure 8-10 Filtering history data by time

The resulting table now contained the exact time period we were interested in, as shown in Figure 8-11.

Worker Threads Summary							
Recording Time	System ID	Gateway Daemon Name	Initial Number	Thread Limit	Current Number Created	Current Number Allocated	Number Of Times Workertimeout Limit Hit
10/15/07 13:15:40	SC66	SCSCT711	100	100	100	100	0
10/15/07 13:20:40	SC66	SCSCT711	100	100	100	100	0
10/15/07 13:25:40	SC66	SCSCT711	100	100	100	100	0
10/15/07 13:30:40	SC66	SCSCT711	100	100	100	87	0
10/15/07 13:35:40	SC66	SCSCT711	100	100	100	100	1
10/15/07 13:40:40	SC66	SCSCT711	100	100	100	100	317
10/15/07 13:45:40	SC66	SCSCT711	100	100	100	100	852
10/15/07 13:50:40	SC66	SCSCT711	100	100	100	100	1342
10/15/07 13:55:40	SC66	SCSCT711	100	100	100	100	2625
10/15/07 14:00:40	SC66	SCSCT711	100	100	100	100	3582
10/15/07 14:05:40	SC66	SCSCT711	100	100	100	100	3993
10/15/07 14:10:40	SC66	SCSCT711	100	100	100	100	4647
10/15/07 14:15:40	SC66	SCSCT711	100	100	100	100	5513
10/15/07 14:20:40	SC66	SCSCT711	100	100	100	100	6125
10/15/07 14:25:40	SC66	SCSCT711	100	100	100	100	6786
10/15/07 14:30:40	SC66	SCSCT711	100	100	100	100	7473
10/15/07 14:35:40	SC66	SCSCT711	100	100	100	100	8137
10/15/07 14:40:40	SC66	SCSCT711	100	100	100	100	8460
10/15/07 14:45:40	SC66	SCSCT711	100	100	100	100	8856
10/15/07 14:50:40	SC66	SCSCT711	100	100	100	100	9479
10/15/07 14:55:40	SC66	SCSCT711	100	100	100	100	9927
10/15/07 15:00:40	SC66	SCSCT711	100	100	100	100	10331
10/15/07 15:05:40	SC66	SCSCT711	100	100	100	100	10597
10/15/07 15:10:40	SC66	SCSCT711	100	100	100	100	10935
10/15/07 15:15:40	SC66	SCSCT711	100	100	100	100	11158
10/15/07 15:20:40	SC66	SCSCT711	100	100	100	32	11309
10/15/07 15:25:40	SC66	SCSCT711	100	100	100	100	11328
10/15/07 15:30:40	SC66	SCSCT711	100	100	100	78	11328
10/15/07 15:35:40	SC66	SCSCT711	100	100	100	71	11328
10/15/07 15:40:40	SC66	SCSCT711	100	100	100	66	11328
10/15/07 15:45:40	SC66	SCSCT711	100	100	100	54	11328
10/15/07 15:50:40	SC66	SCSCT711	100	100	100	32	11328

Figure 8-11 Worker thread statistics filtered for the time period of the test

We repeated this process for three Gateway daemon summary tables:

- ▶ Worker thread Summary (Figure 8-11 on page 314)
- ▶ Connection Manager Summary (Figure 8-12)
- ▶ Transaction Analysis Summary (Figure 8-13 on page 316)

Connection Manager Threads Summary								
Recording Time	System ID	Gateway Daemon Name	Initial Number	Thread Limit	Current Number Created	Current Number Allocated	Number Of Times Connecttimeout Limit Hit	Number Waiting
10/15/07 13:15:35	SC66	SCSCT711	500	500	500	300	0	0
10/15/07 13:20:35	SC66	SCSCT711	500	500	500	300	0	0
10/15/07 13:25:35	SC66	SCSCT711	500	500	500	300	0	11
10/15/07 13:30:35	SC66	SCSCT711	500	500	500	300	0	4
10/15/07 13:35:35	SC66	SCSCT711	500	500	500	400	0	84
10/15/07 13:40:35	SC66	SCSCT711	500	500	500	400	0	143
10/15/07 13:45:35	SC66	SCSCT711	500	500	500	400	0	191
10/15/07 13:50:35	SC66	SCSCT711	500	500	500	400	0	159
10/15/07 13:55:35	SC66	SCSCT711	500	500	500	400	0	184
10/15/07 14:00:35	SC66	SCSCT711	500	500	500	400	0	142
10/15/07 14:05:35	SC66	SCSCT711	500	500	500	400	0	170
10/15/07 14:10:35	SC66	SCSCT711	500	500	500	400	0	133
10/15/07 14:15:35	SC66	SCSCT711	500	500	500	400	0	106
10/15/07 14:20:35	SC66	SCSCT711	500	500	500	400	0	184
10/15/07 14:25:35	SC66	SCSCT711	500	500	500	400	0	140
10/15/07 14:30:35	SC66	SCSCT711	500	500	500	400	0	137
10/15/07 14:35:35	SC66	SCSCT711	500	500	500	400	0	156
10/15/07 14:40:35	SC66	SCSCT711	500	500	500	400	0	113
10/15/07 14:45:35	SC66	SCSCT711	500	500	500	400	0	175
10/15/07 14:50:35	SC66	SCSCT711	500	500	500	400	0	175
10/15/07 14:55:35	SC66	SCSCT711	500	500	500	400	0	156
10/15/07 15:00:35	SC66	SCSCT711	500	500	500	400	0	137
10/15/07 15:05:35	SC66	SCSCT711	500	500	500	400	0	176
10/15/07 15:10:35	SC66	SCSCT711	500	500	500	400	0	21
10/15/07 15:15:35	SC66	SCSCT711	500	500	500	400	0	20
10/15/07 15:20:35	SC66	SCSCT711	500	500	500	400	0	148
10/15/07 15:25:35	SC66	SCSCT711	500	500	500	400	0	0
10/15/07 15:30:35	SC66	SCSCT711	500	500	500	400	0	0
10/15/07 15:35:35	SC66	SCSCT711	500	500	500	383	0	0
10/15/07 15:40:35	SC66	SCSCT711	500	500	500	300	0	0
10/15/07 15:45:35	SC66	SCSCT711	500	500	500	300	0	0

Figure 8-12 Filtered Connection Manager summary

Gateway Daemon Summary							
Recording Time	System ID	Gateway Daemon Name	Number Of Requests Processed	Successful SYNCONRETURN Transactions	Extended LUW Transactions Committed	Extended LUW Transactions Rolled Back	XA Transactions Committed
10/15/07 13:15:38	SC66	SC8CT711	3900	4100	0	0	0
10/15/07 13:20:38	SC66	SC8CT711	62756	62956	0	0	0
10/15/07 13:25:38	SC66	SC8CT711	121635	121833	0	0	0
10/15/07 13:30:38	SC66	SC8CT711	180347	180541	0	0	0
10/15/07 13:35:38	SC66	SC8CT711	238897	238897	0	0	0
10/15/07 13:40:38	SC66	SC8CT711	295415	295615	0	0	0
10/15/07 13:45:38	SC66	SC8CT711	352598	352798	0	0	0
10/15/07 13:50:38	SC66	SC8CT711	409260	409460	0	0	0
10/15/07 13:55:38	SC66	SC8CT711	467809	467809	0	0	0
10/15/07 14:00:38	SC66	SC8CT711	525446	525646	0	0	0
10/15/07 14:05:38	SC66	SC8CT711	581788	581788	0	0	0
10/15/07 14:10:38	SC66	SC8CT711	640741	640741	0	0	0
10/15/07 14:15:38	SC66	SC8CT711	698961	699061	0	0	0
10/15/07 14:20:38	SC66	SC8CT711	756596	756596	0	0	0
10/15/07 14:25:38	SC66	SC8CT711	814421	814621	0	0	0
10/15/07 14:30:38	SC66	SC8CT711	872667	872667	0	0	0
10/15/07 14:35:38	SC66	SC8CT711	930800	930999	0	0	0
10/15/07 14:40:38	SC66	SC8CT711	988041	988241	0	0	0
10/15/07 14:45:38	SC66	SC8CT711	1045848	1046048	0	0	0
10/15/07 14:50:38	SC66	SC8CT711	1105282	1105282	0	0	0
10/15/07 14:55:38	SC66	SC8CT711	1163875	1164075	0	0	0
10/15/07 15:00:38	SC66	SC8CT711	1222292	1222292	0	0	0
10/15/07 15:05:38	SC66	SC8CT711	1280346	1280545	0	0	0
10/15/07 15:10:38	SC66	SC8CT711	1339569	1339769	0	0	0
10/15/07 15:15:38	SC66	SC8CT711	1397313	1397511	0	0	0
10/15/07 15:20:38	SC66	SC8CT711	1452379	1452577	0	0	0
10/15/07 15:25:38	SC66	SC8CT711	1505120	1505237	0	0	0
10/15/07 15:30:38	SC66	SC8CT711	1553535	1553696	0	0	0
10/15/07 15:35:38	SC66	SC8CT711	1592594	1592694	0	0	0
10/15/07 15:40:38	SC66	SC8CT711	1633766	1633898	0	0	0
10/15/07 15:45:38	SC66	SC8CT711	1671887	1671887	0	0	0

Figure 8-13 Filtered Transaction Analysis summary

We now had the information filtered for the time period of the test and in a form where we could see a detailed picture of the various Gateway daemon components. However, the columns in the tables are the default columns provided with the view. There are more statistics we could include in these tables. To include these *hidden* columns, the filter properties of the table have to be changed. To do this:

1. Right-click a row in the table.
2. Select **Properties**. The Gateway daemon Resources window appears, as shown in Figure 8-14 on page 317.
3. Select the **Filters** tab.
4. Tick or untick the data items you want included or excluded.
5. Select **OK**.

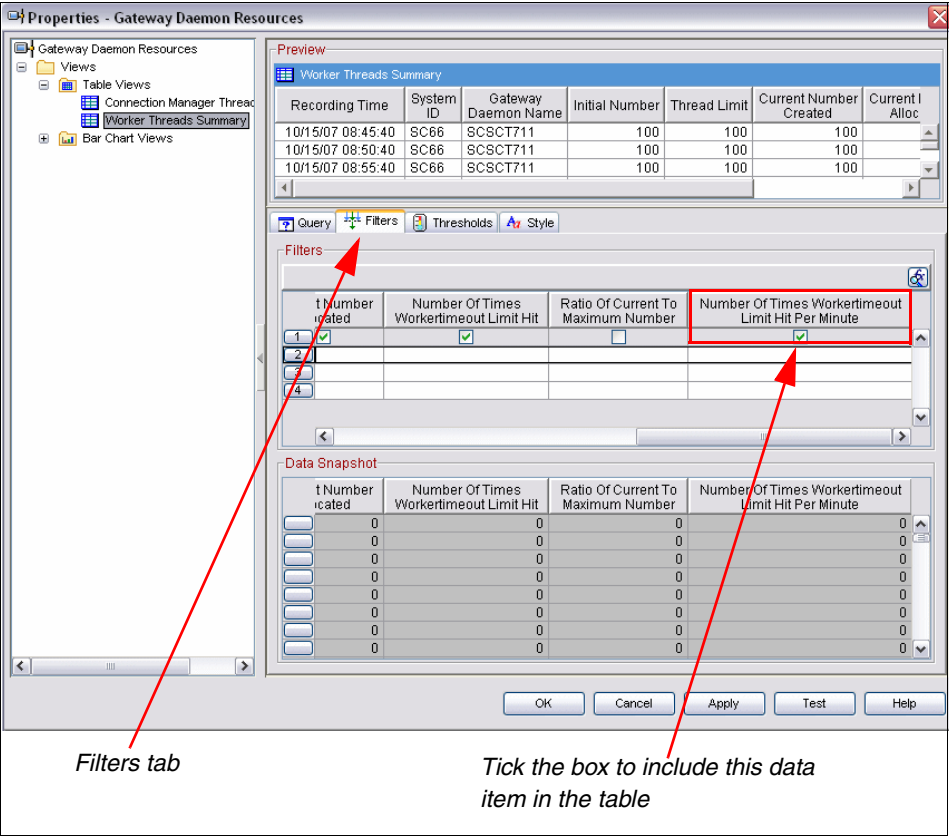


Figure 8-14 Properties - Gateway daemon Resources

We ticked the **Number Of Times Workertimeout Limit Hit Per Minute** box and discovered that the summary table had changed to include the new column, as shown in Figure 8-15.

Recording Time	System ID	Gateway Daemon Name	Initial Number	Thread Limit	Current Number Created	Current Number Allocated	Number Of Times Workertimeout Limit Hit	Number Of Times Workertimeout Limit Hit Per Minute
10/15/07 13:00:40	SC66	SCSCT711	100	100	100	0	0	0
10/15/07 13:05:40	SC66	SCSCT711	100	100	100	0	0	0
10/15/07 13:15:40	SC66	SCSCT711	100	100	100	0	0	0
10/15/07 13:20:40	SC66	SCSCT711	100	100	100	0	0	0
10/15/07 13:25:40	SC66	SCSCT711	100	100	100	0	0	0
10/15/07 13:30:40	SC66	SCSCT711	100	100	100	87	0	0
10/15/07 13:35:40	SC66	SCSCT711	100	100	100	100	1	0
10/15/07 13:40:40	SC66	SCSCT711	100	100	100	100	317	109
10/15/07 13:45:40	SC66	SCSCT711	100	100	100	100	852	171
10/15/07 13:50:40	SC66	SCSCT711	100	100	100	100	1342	149
10/15/07 13:55:40	SC66	SCSCT711	100	100	100	100	2625	328
10/15/07 14:00:40	SC66	SCSCT711	100	100	100	100	3582	307
10/15/07 14:05:40	SC66	SCSCT711	100	100	100	100	3993	173
10/15/07 14:10:40	SC66	SCSCT711	100	100	100	100	4647	213

Figure 8-15 Table including hidden statistics

We also configured the TEP client to show some of the summary data in a graphical form. To do this task, we went back to the Select the time span view shown in Figure 8-10 on page 313, ticked the **Apply to all views associated with this view's query**, and then selected **OK**. We did this for both of the Gateway daemon Resources views. The resulting chart represents the Worker and Connection Manager Threads summary for the problem period, as shown in Figure 8-16.

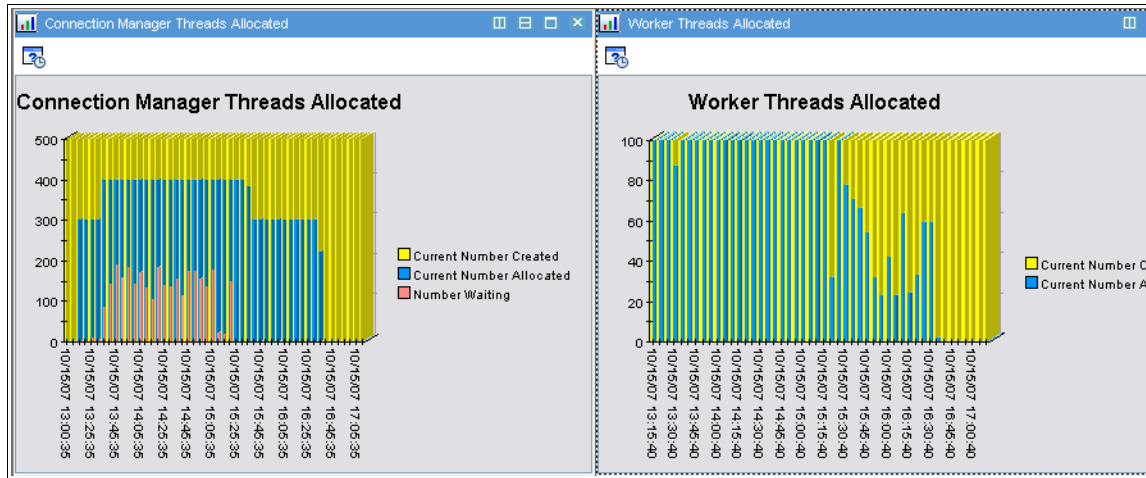


Figure 8-16 Worker and Connection Manager summaries expressed in graphical form

The data included in the chart can be altered by changing its properties to include or excluded columns. To do this, do the following steps:

1. Right-click the chart.
2. Selected **Properties**. The Gateway daemon Resources window appears, as shown in Figure 8-14 on page 317.
3. Select the **Filters** tab.
4. Tick or untick the data items you want included or exclude.
5. Select **OK**.

The format of the chart itself can be changed by clicking a different chart type on the toolbar, as shown in Figure 8-17 on page 319.

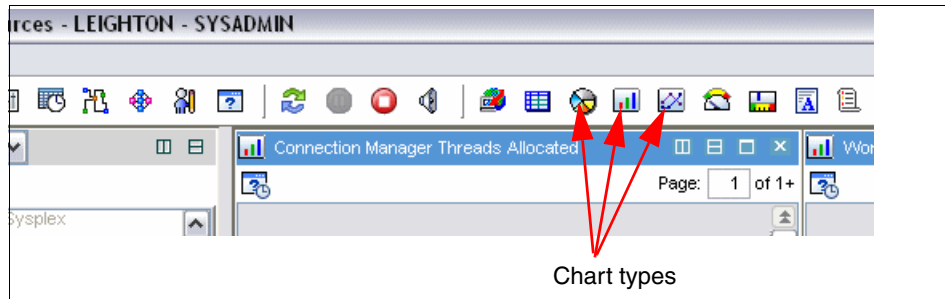


Figure 8-17 Chart types on the toolbar

We changed the chart type to a line chart. Finally, we closed one of the chart views and dragged the remaining chart to a larger size. The resulting chart can be seen in Figure 8-18.

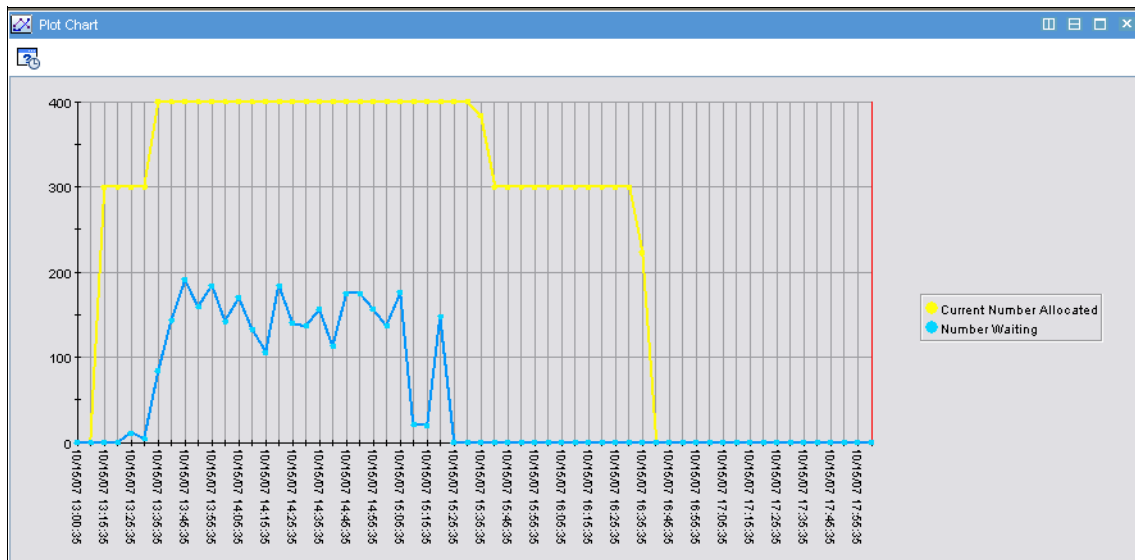


Figure 8-18 Chart of the Connection Manager thread activity during the test

We examined the three summary tables in Figure 8-11 on page 314, Figure 8-12 on page 315, and Figure 8-13 on page 316, and created a table cross-referencing the key statistics collected every five minutes (see Table 8-6).

Table 8-6 Key statistics collated from the summary tables

Time	Requests	Request TPS^a	WT Allocates	WT Timeouts	WT (int) Timeouts^b	CM Threads	CM Waiters
13:15:38	3900		100			300	0
13:20:38	62756	195	100			300	0
13:25:38	121635	196	100			300	11
13:30:38	180347	195	87			300	4
13:35:38	238897	195	100	1	1	400	84
13:40:38	295415	188	100	317	316	400	143
13:45:38	352598	190	100	852	535	400	191
13:50:38	409260	188	100	1342	490	400	159
13:55:38	467809	195	100	2625	1283	400	184
14:00:38	525446	192	100	3582	957	400	142
14:05:38	581788	187	100	3993	411	400	170
14:10:38	640741	196	100	4647	654	400	133
14:15:38	698861	193	100	5513	866	400	106
14:20:38	756596	192	100	6125	612	400	184
14:25:38	814421	192	100	6786	661	400	140
14:30:38	872667	194	100	7473	687	400	137
14:35:38	930800	194	100	8137	664	400	156
14:40:38	988041	190	100	8460	323	400	113
14:45:38	1045848	192	100	8856	396	400	175
14:50:38	1105282	198	100	9479	623	400	175
14:55:38	1163875	195	100	9927	448	400	156
15:00:38	1222292	194	100	10331	404	400	137
15:05:38	1280346	193	100	10597	266	400	176

15:10:38	1339569	197	100	10935	338	400	21
15:15:38	1397313	192	100	11158	223	400	20
15:20:38	1452379	183	32	11309	151	400	148
15:25:38	1505120	175	100	11328	19	400	0
15:30:38	1553535	161	78	11328		400	0
15:35:38	1592594	130	71	11328		383	0
15:40:38	1633766	137	66	11328		300	0

- a. Calculated as TPS from interval delta
- b. Calculated as interval difference from lifelong totals

We took stats from the columns Requests (TPS), WT timeouts, CManager allocation, and Connection Manager waiters and created a chart to aid analysis, as shown in Figure 8-19.

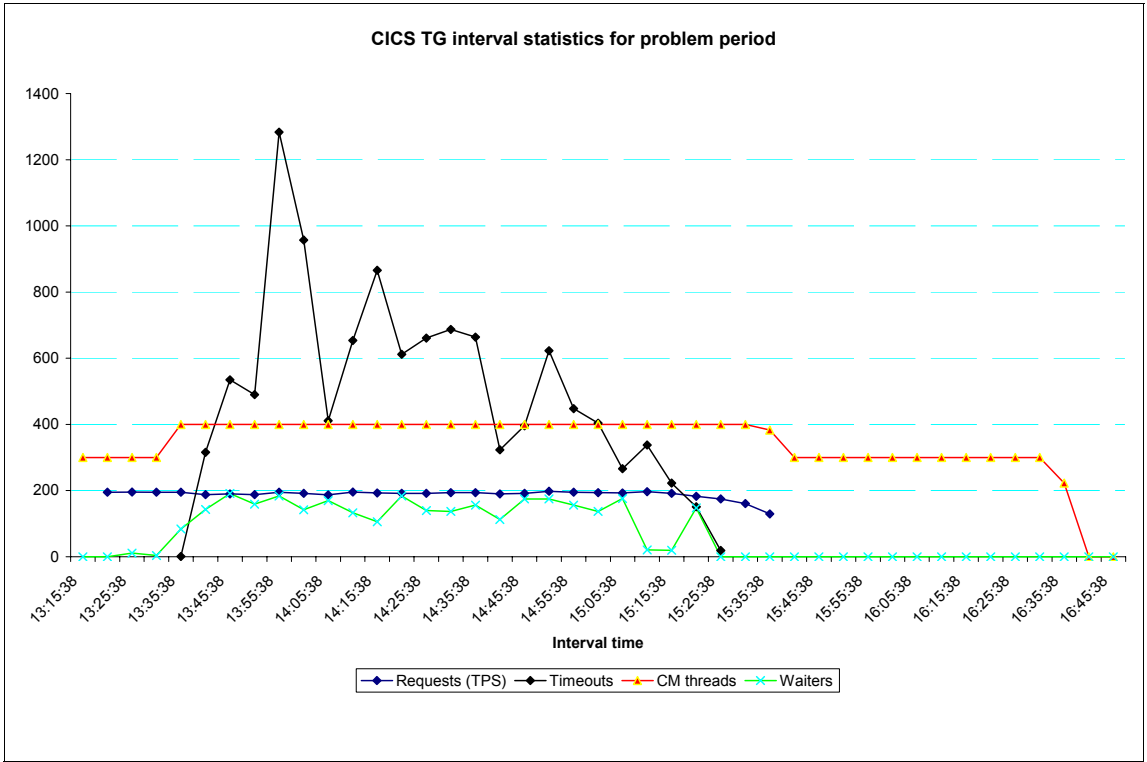


Figure 8-19 Chart of the interval statistics extracted from TEP summary tables

From these statistics, we worked out the timeline of the base workload and the time of the extra workload in relation to the SMF statistic intervals, as shown in Figure 8-20.

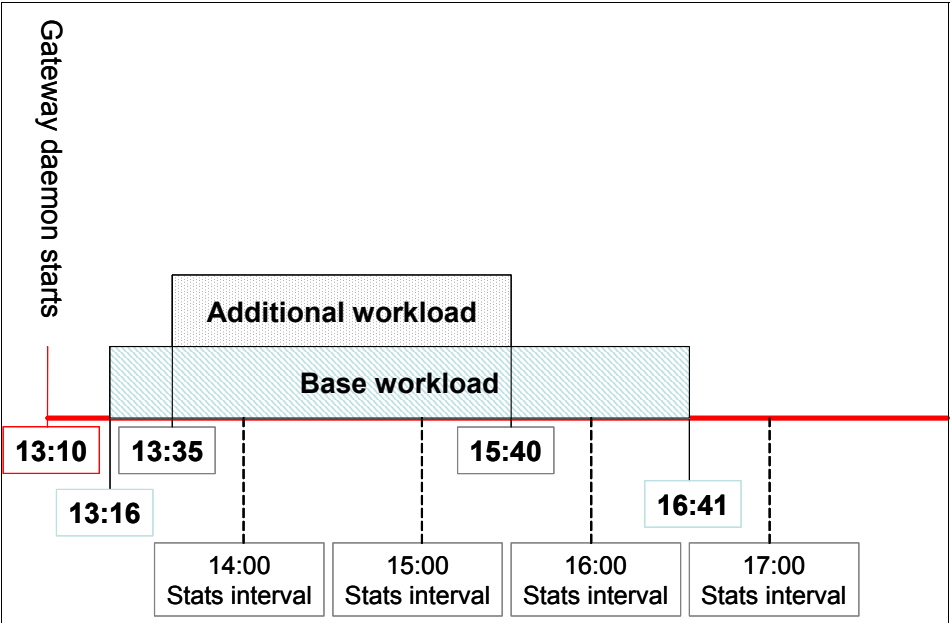


Figure 8-20 Timeline of the scenario

We had the following observations about the interval statistics collated in Table 8-6 on page 320:

Current Number Allocated

The number of allocated connection managers increases at 13:35 (from 300 to 400) and returns back to 300 at 15:30. This period (13:35-15:30) appears to be when the extra workload is present.

Number Waiting

When the number of Current Number Allocated increases, we have an increase in totals in both the Number Waiting and Number Of Times Workertimeout Limit Hit columns, indicating that we have a backlog of calls in the Gateway daemon.

Number Waiting

The average number of Number Waiting in the Gateway daemon based on these totals is 138. This represents the backlog in the Gateway daemon of waiting requests.

Number Of Times Workertimeout Limit Hit

Connection Manager requests that a wait longer than the workertimeout value of 1000 ms will timeout. The number of timeouts increases gradually over the period 13:35 - 15:30, so these timeouts are not caused by a single peak in workload but by a gradual trend.

Number Of Requests Processed

This category stayed constantly high during the period of extra workload. An average TPS during the first hour (13:35 - 14:35) is 192, but during the second hour (14:35 - 15:35) it drops to 183.

CICS response time CICS and Gateway daemon response time stats are currently not collected by OMEGAMON XE. SMF stats for intervals between 14:00 - 16:00 show CICS response times of 515 ms (194 TPS). The CICS region is still operating at near full capacity.

Using the stats collated from both SMF and OMEGAMON XE, we can calculate the new workload, based on the base workload plus the extra workload, as follows:

- ▶ Referring to the SMF interval 2 in Table 8-5 on page 306, we have a Gateway daemon average response time of 1174 ms and a CICS response time of 515 ms.
- ▶ Referring to Table 8-6 on page 320, the base workload Connection Manager threads is 300 and the additional workload value is 100.
- ▶ Think time for the base workload is 1.0 seconds.
- ▶ Think time for the additional workload is 0.1 second.
- ▶ Maximum workload driver rate: Number of users / (thinktime + RespTime), where:
 - For the base workload: $300 / (1.0 + 1.174) = 138$ TPS
 - For the additional workload: $100 / (0.1 + 1.174) = 78$
 - For the combined workload during interval 2: $138 + 85 = 216$ TPS
- ▶ The CICS average response time is 0.515 seconds, which equates to 194 TPS.

The problem is that the Gateway daemon and CICS region configuration cannot cope with the extra workload. Connection Manager requests are queued in the Gateway daemon waiting for a Worker thread and those that wait longer than one second are being timed out.

8.3.7 Action taken

To handle the extra workload without the Connection Manager, including waits and worker timeouts, we need to have more Worker threads. How many Worker threads do we need?

We can go back to our equation and calculate the number of Worker threads needed to handle the new workload to achieve the required TPS. We calculated the number of Worker threads based on a CICS response time of 500 ms as follows:

numW = number of Worker threads, *TPS* = transactions per second, *CICS RespTime* = average response time through the CICS region, *numUsers* = number of concurrent users, and *thinkTm* = average user think time.

$$numW = CICS\ respTime \times numUsers / (thinkTm + CICS\ respTime).$$

The above formulas are intended as a very rough general rule. All systems have different dynamics, so these formulas should be used as a starting point for tuning the system only. All our timings are in seconds.

Additional workload: $0.5 \times 300 / (1.0 + 0.5) = 100$ Worker threads

Additional workload: $0.5 \times 100 / (0.1 + 0.5) = 83$ Worker threads

Therefore, in theory, we need 183 Worker threads to achieve 200 TPS on the CICS region with an average 0.5 second response time.

We changed maxworker in the Gateway daemon configuration file (ctg.ini) to increase the number of Worker threads. To implement the new configuration file, we recycle SCST711. At startup, we received the message shown in on the STDOUT of the Gateway daemon.

Example 8-2 Message issued when maximum Worker threads exceeds the EXCI limit

CTG6406W The maximum Worker threads has been reduced to the EXCI logon limit 100.

We attempted to use more than the limit of Worker threads. A single z/OS address space is limited by CICS interregion communication (IRC) to allocating a maximum number of EXCI pipes for all attached CICS regions. You can allocate up to 250 pipes in an EXCI address space (when APAR PQ92943 is applied); the default limit is 100 pipes. This value can be changed by specifying LOGONLIM in member DFHSSIyy of the SYS1.PARMLIB library. For more information about how to specify the LOGONLIM parameter, see the *EXCI Pipe Allocation* topic on the CICS TS InfoCenter.

We did not want to change our configuration, so we chose to allocate the necessary Worker threads by starting a second Gateway daemon when an increase in workload causes CM waits and timeouts. As per Figure 8-1 on page 298, our configuration is a dual Gateway daemon and CICS region with TCP/IP port sharing. Once started, the second Gateway daemon (SCSCT714) will handle half the workload by providing an extra 100 Worker threads.

OMEGAMON XE was configured to automate the start of the second Gateway daemon using an alert (Situation) when the following conditions occurred:

- ▶ The Connection Manager requests waiting for threads reached 100.
- ▶ The Worker thread timeouts reached 150.

When these conditions occurred, a Situation would be raised and an action would be taken to start another Gateway daemon.

We created a Situation using the TEP client as follows:

1. We selected the CICS TG node for the SCSCT711 on the navigator view.
2. We right-clicked the instance and selected **Situations**, as shown in Figure 8-21.

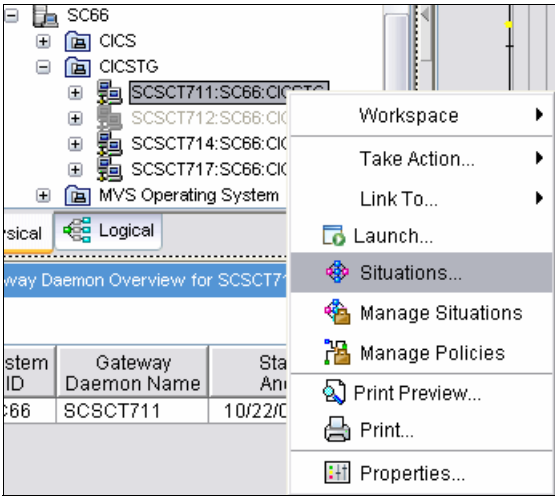


Figure 8-21 Selecting the situation editor

3. The situation editor pop-up menu appeared. We right-clicked and then selected Create New, as shown in Figure 8-22.

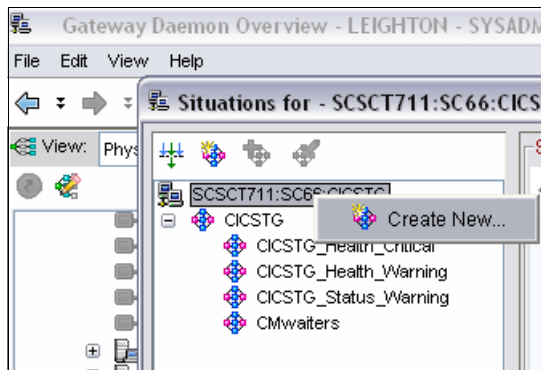


Figure 8-22 Select CREATE NEW

4. In the Create Situation window, we entered the name and details of the Situation and pressed **OK**, as shown in Figure 8-23. The Situation (CMwaiters) was created and its name appeared in the left hand frame.

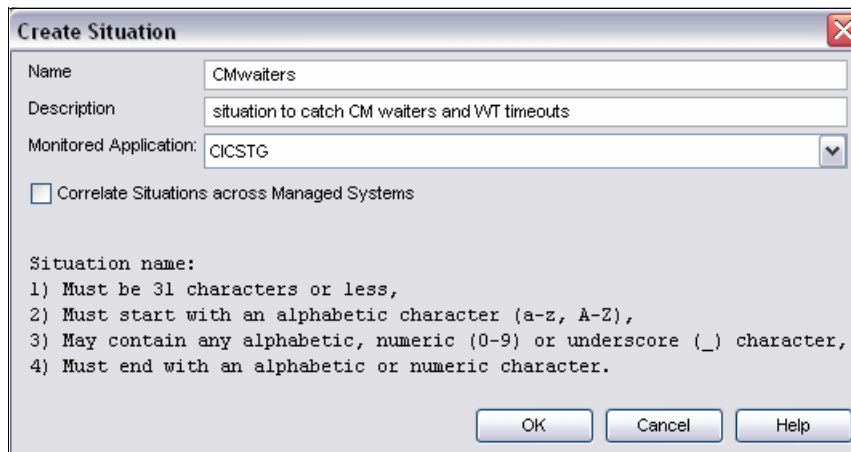


Figure 8-23 Enter details of the situation

5. A further window appeared where “conditions” can be specified. We selected **CICSTG Connection Manager Threads** in the left hand box (attribute group). The selection changed in the right hand box (attribute item), where we then selected **Number Waiting** from the attribute item list and clicked **OK**, as shown in Figure 8-24 on page 327.

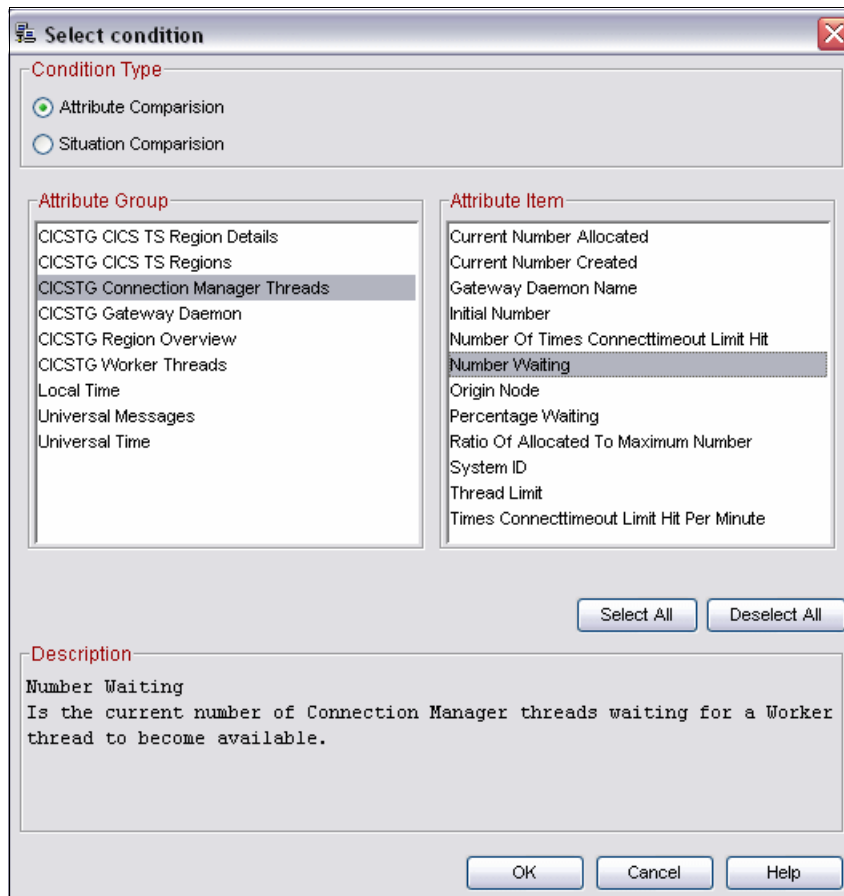


Figure 8-24 Select situation conditions

- The situation editor view will now appear showing the condition we had selected, as shown in Figure 8-26 on page 329. This can be used to add further conditions. We wanted two conditions with values assigned, so we added a condition for CICSTG Worker threads: Number Of Times Workertimout Limit Hit. We then assigned values using the formula bar:
 - Number Waiting => 100
 - Number Of Times Workertimout Limit Hit => 150
- The sampling interval is the amount of time between checks on the status of the conditions. We set this to five minutes.

8. We wanted the Situation to start a second Gateway daemon when the conditions were met. To achieve this goal, we created an action by selecting the **Action** tab (see Figure 8-25). The system command to start the Gateway daemon was the modify command S SCSCT714.

The screenshot shows a software window with a tabbed interface. The 'Action' tab is selected. The window contains several sections with radio button options:

- Action Selection:** ☒ System Command, ☐ Universal Message
- System Command:** A text field containing 'S SCSCT714' and a button labeled 'Attribute Substitution...'
- If the condition is true for more than one monitored item:** ☒ Only take action on first item, ☐ Take action on each item
- Where should the Action be executed (performed):** ☒ Execute the Action at the Managed System (Agent), ☐ Execute the Action at the Managing System (TMS)
- If the condition stays true over multiple intervals:** ☒ Don't take action twice in a row (wait until situation goes false then true again), ☐ Take action in each interval

At the bottom of the window are four buttons: 'OK', 'Cancel', 'Apply', and 'Help'.

Figure 8-25 Setting an action for the Situation

Attention: When setting a system command, do not prefix it with a /.

9. We finished by pressing **OK**, and the situation was established (see Figure 8-26).

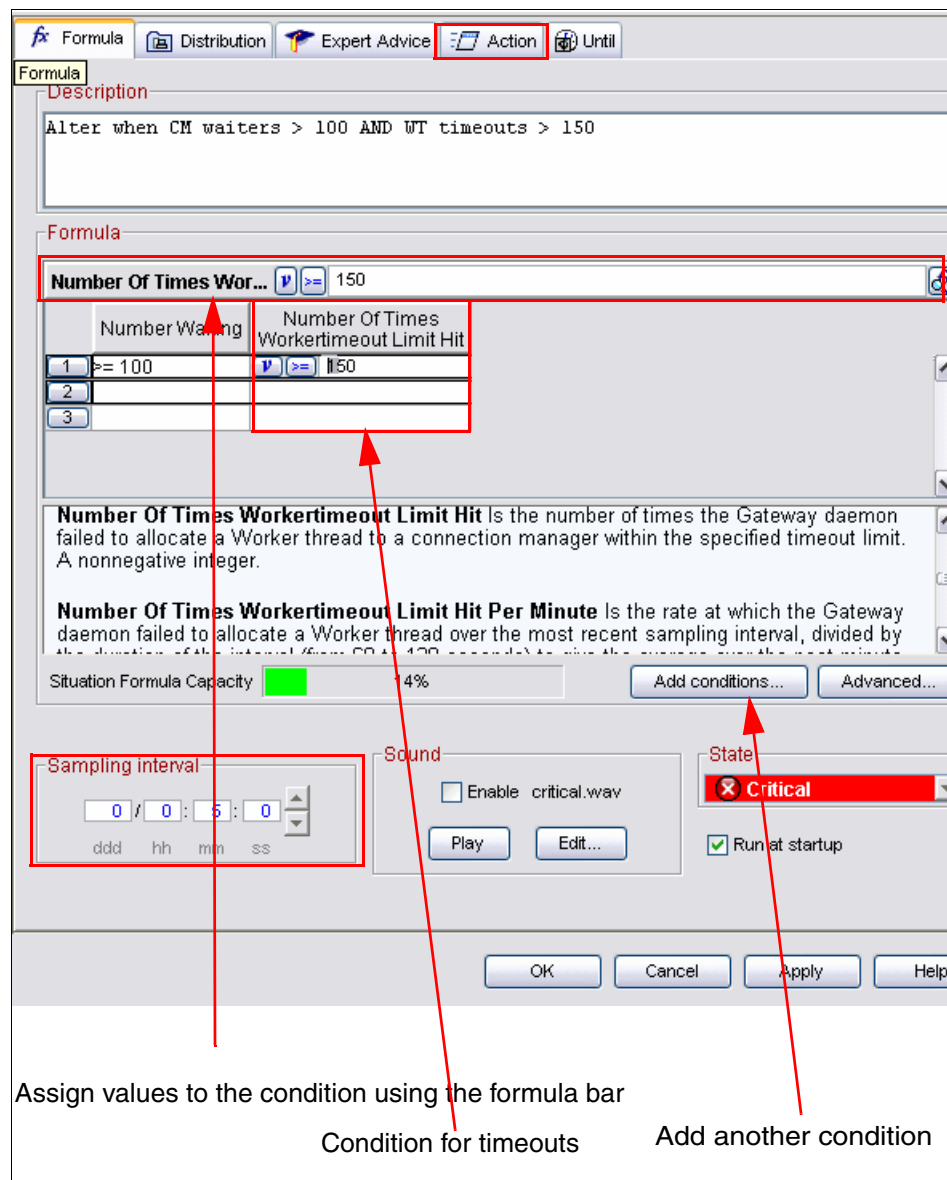


Figure 8-26 Adding conditions and assigning values

To check if the situation has been successfully created, go back to the navigation window and right-click the SCST711 workspace, and then select **Manage Situations** (see Figure 8-27) to display the situations being managed.

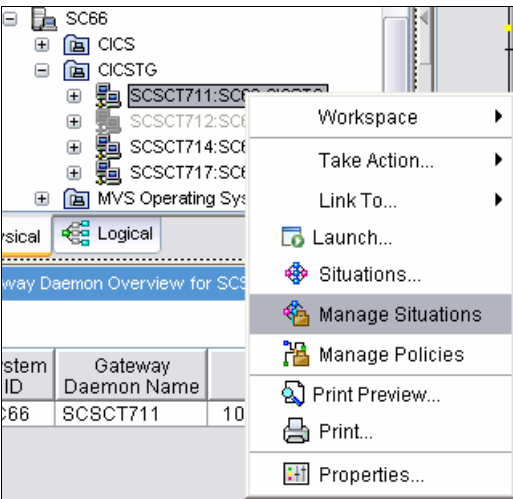


Figure 8-27 Select Manage Situations

The Manage Situations window confirms that the situation CMwaiters has been created and is currently started, as shown in Figure 8-28 on page 331.

Manage Situation at Managed System: SCST711:SC66:CICSTG		
Name	Status	Description
CICSTG_CommFailure_Warning	Closed	CICS TG CICS Communication Failure Warning
CICSTG_ConnAlloc_Warning	Started	CICS TG Connection Manager Thread Allocation War
CICSTG_ConnTimeout_Warning	Closed	CICS TG Connection Manager Threads Timeout War
CICSTG_ConnWait_Warning	Closed	CICS TG Connection Manager Threads Waiting Warr
CICSTG_Freepipe_Warning	Stopped	CICS TG Freepipe Warning
CICSTG_Health_Critical	Closed	CICS TG Health Critical
CICSTG_Health_Warning	Closed	CICS TG Health Warning
CICSTG_PipeAlloc_Warning	Started	CICS TG EXCI Pipe Allocation Failure Warning
CICSTG_RollbackLUW_Critical	Closed	CICS TG Rollback LUW Critical
CICSTG_RollbackLUW_Warning	Started	CICS TG Rollback LUW Warning
CICSTG_RollbackXA_Critical	Closed	CICS TG Rollback XA Critical
CICSTG_RollbackXA_Warning	Stopped	CICS TG Rollback XA Warning
CICSTG_Status_Warning	Closed	CICS TG Status Warning
CICSTG_WorkerAlloc_Warning	Closed	CICS TG Worker Thread Allocation Warning
CICSTG_WorkerTimeout_Warning	Closed	CICS TG Worker Threads Timeout Warning
CMwaiters	Started	Alter when CM waiters > 100 AND WT timeouts > 150
MissingCICSTG	Open	alert when a CICSTG goes down
StatsCT711	Open	When turned on will display CTG stats on each interv
XA_health_reset	Started	Resetting CICS TG health to 100

Figure 8-28 Manage Situations display showing CMwaiters started

The base workload was rerun using the same configuration described in 8.3.1, “Creating the base workload” on page 298.

After 20 minutes, the workload changed. As before, Current Number Allocated increased from 300 to 400. Number Waiting and Number Of Times Workertimeout Limit Hit began to increase. Subsequently, the SCST711 node was flagged as critical; a small red circle with a X appeared on the node. Hovering the mouse over the SCST711 node presents a pop-up of the Gateway daemon’s Situation status. Figure 8-29 on page 332 shows the current situations; there is a CRITICAL alarm for situation CMwaiters at 09:13:12.

From the same figure, we saw that the real time values for Number Waiting is 139 and Number Of Times workertimeout Limit Hit is 855. Therefore, the conditions for CMwaiters has been met, causing the situation to trigger.

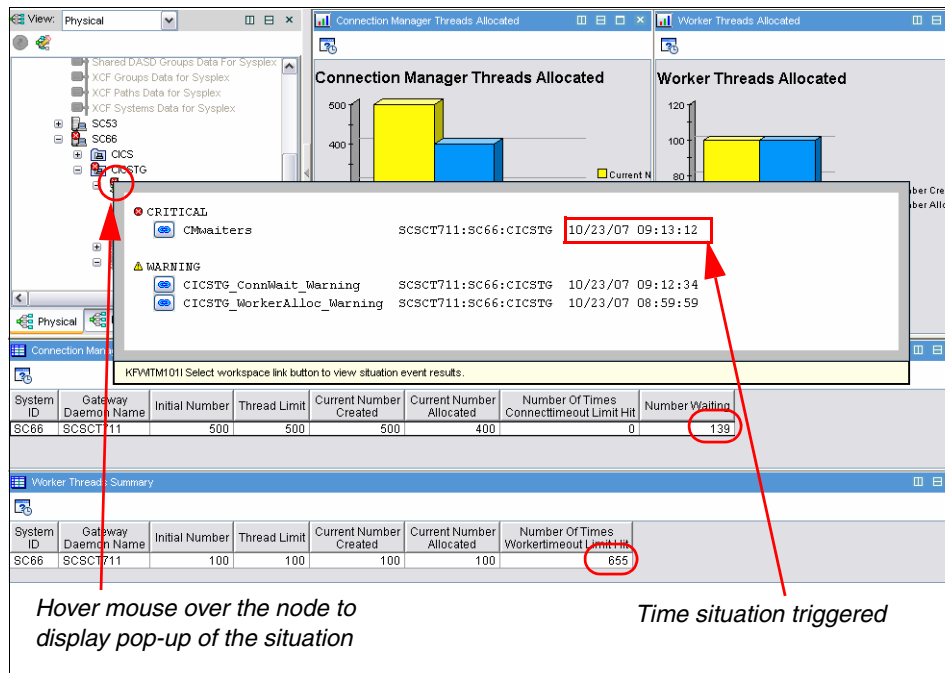


Figure 8-29 Situation CMwaiters triggered a critical alert

Looking at the SDSF log at 09:13:12, we found a modify command (see Example 8-3) that started SCST714.

Example 8-3 Modify command issued by the situation CMwaiters to start SCST714

```
09:13:12.03 STC01536 00000290 S SCST714
09:13:12.09 STC09482 00000090 $HASP100 SCST714 ON STCINRDR
09:13:12.16 STC09482 00000290 IEF695I START SCST714 WITH JOBNAME
SCST714 IS ASSIGNED TO USER
CTGUSER , GROUP CICS
```

The Situation Event Console can be displayed by clicking the top most node (Enterprise) in the navigator view. This shows that a critical situation exists for CMwaiters, as shown in Figure 8-30 on page 333.

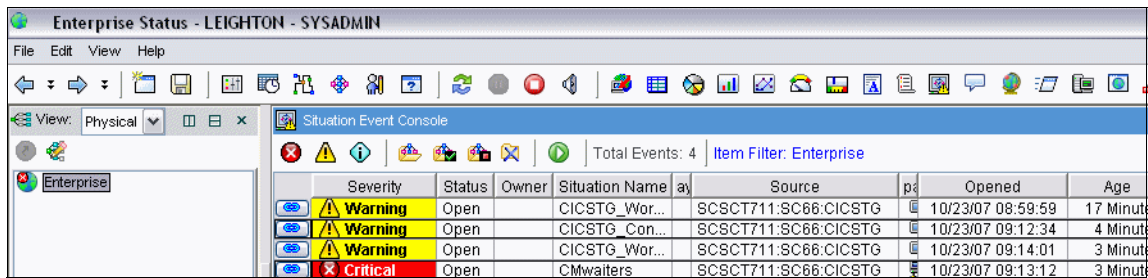


Figure 8-30 Situation Event Console

Finally, after SCSCT714 had been started by the situation CMwaiter, historical statistics were used to confirm that the workload was being shared across the two Gateway daemons. Figure 8-31 shows that the number of Connection Manager threads dropped from 400 to 196. Also, Number Waiting dropped down to single figures and Number Of Times Workertimeout Limit Hit remains constant. These statistics indicate that the two Gateway daemons are coping with the new workload.

Recording Time	System ID	Gateway Daemon Name	Initial Number	Thread Limit	Current Number Created	Current Number Allocated	Number Of Times Connecttimeout Limit Hit	Number Waiting
10/23/07 08:50:00	SC66	SCSCT711	500	500	500	300	0	0
10/23/07 08:55:00	SC66	SCSCT711	500	500	500	300	0	8
10/23/07 09:00:00	SC66	SCSCT711	500	500	500	300	0	0
10/23/07 09:05:00	SC66	SCSCT711	500	500	500	300	0	24
10/23/07 09:10:00	SC66	SCSCT711	500	500	500	400	0	65
10/23/07 09:15:00	SC66	SCSCT711	500	500	500	400	0	149
10/23/07 09:20:00	SC66	SCSCT711	500	500	500	196	0	0
10/23/07 09:25:00	SC66	SCSCT711	500	500	500	196	0	0
10/23/07 09:30:00	SC66	SCSCT711	500	500	500	196	0	0
10/23/07 09:35:00	SC66	SCSCT711	500	500	500	196	0	1
10/23/07 09:40:00	SC66	SCSCT711	500	500	500	196	0	0
10/23/07 09:45:00	SC66	SCSCT711	500	500	500	196	0	2
10/23/07 09:50:00	SC66	SCSCT711	500	500	500	196	0	3
10/23/07 09:55:00	SC66	SCSCT711	500	500	500	196	0	0
10/23/07 10:00:00	SC66	SCSCT711	500	500	500	196	0	0
10/23/07 10:05:00	SC66	SCSCT711	500	500	500	196	0	0
10/23/07 10:10:00	SC66	SCSCT711	500	500	500	196	0	0
10/23/07 10:15:00	SC66	SCSCT711	500	500	500	196	0	0

Figure 8-31 Connection Manager summary showing relief after the alert

8.4 Using CICS PA to analyze CICS TG SMF historical data

IBM CICS Performance Analyzer on z/OS (CICS PA) V2.1 recently introduced support for analyzing the statistics that CICS TG writes to SMF type 111 records. CICS PA introduced this support in the new function APAR PK53163 (PTFs UK31800, UK31810, UK31818, and UK31797).

This section describes how you can use CICS PA to solve problems in CICS TG.

8.4.1 What CICS PA is

CICS PA is a reporting tool that provides information about the performance of your CICS systems and applications to help you tune, manage, and plan your CICS systems effectively.

CICS PA is not an online monitoring tool. It produces reports and extracts using data collected by your system in MVS System Management Facility (SMF) data sets, which include:

- ▶ CICS Monitoring Facility (CMF) performance class, exception class, and transaction resource class data in SMF 110 records
- ▶ CICS Transaction Server statistics data in SMF 110 records
- ▶ CICS Transaction Gateway statistics data in SMF 111 records
- ▶ System Logger data in SMF 88 records
- ▶ DB2 accounting data in SMF 101 records
- ▶ WebSphere MQ accounting data in SMF 116 records
- ▶ IBM Tivoli OMEGAMON XE for CICS on z/OS (OMEGAMON XE for CICS) data in SMF 112 records, containing transaction data for Adabas, CA-Datcom, CA-IDMS, and Supra database management systems

CICS PA can help:

- ▶ System Programmers track the overall CICS system performance and evaluate the results of their system tuning efforts
- ▶ Application Programmers analyze the performance of their applications and the resources they use
- ▶ Database Administrators analyze the usage and performance of database systems such as IMS™ and DB2
- ▶ MQ Administrators analyze the usage and performance of their WebSphere MQ messaging systems
- ▶ Managers ensure transactions are meeting their required Service Levels and measure trends to help plan future requirements and strategies

CICS PA reports all aspects of CICS system activity and resource usage, including:

- ▶ Transaction response time
- ▶ CICS system resource usage
- ▶ Cross-system performance, including multi-region operation (MRO) and advanced program-to-program communication (APPC)

- ▶ CICS Business Transaction Services (BTS)
- ▶ CICS Web Support
- ▶ External subsystems, including DB2, IMS, and WebSphere MQ
- ▶ System Logger performance
- ▶ Exception events that cause performance degradation
- ▶ Transaction file and temporary storage usage

Rather than keeping large SMF data sets for reporting purposes, you can use CICS PA to load selected SMF records into a CICS PA historical database (HDB), optionally summarizing the records according to the time intervals that you require for reporting (such as hourly or daily). You can then use CICS PA to produce reports from the HDB instead of the SMF data sets. Loading selected and summarized SMF data into an HDB allows you to accumulate the performance data you want at the level of detail you need for reporting over long periods, without requiring large amounts of storage or processing time.

In addition to producing formatted reports from SMF data sets or HDBs, CICS PA can extract data to DB2 tables or comma-separated value (CSV) text files. You can then develop your own custom reports using DB2 SQL queries, or download CSV files to your PC, where you can view and manipulate the data using PC-based spreadsheet applications such as Microsoft Excel®.

8.4.2 CICS PA support for analyzing CICS TG SMF 111 records

CICS PA provides the following support for analyzing CICS TG SMF 111 records:

- ▶ The CICS PA ISPF dialog presents a formatted display of CICS TG statistics, either from an SMF data set or a CICS PA HDB, with online help describing each statistics field. You can save the formatted displays to a text file.
- ▶ For long-term reporting, you can collect SMF 111 records in a CICS PA HDB.
- ▶ You can export data from CICS PA HDBs to either comma-separated value (CSV) files, for use with applications such as PC-based spreadsheet software, or DB2 tables, so that you can use SQL queries to create custom reports.

Figure 8-32 gives an overview of support for CICS TF SMF 111 statistics.

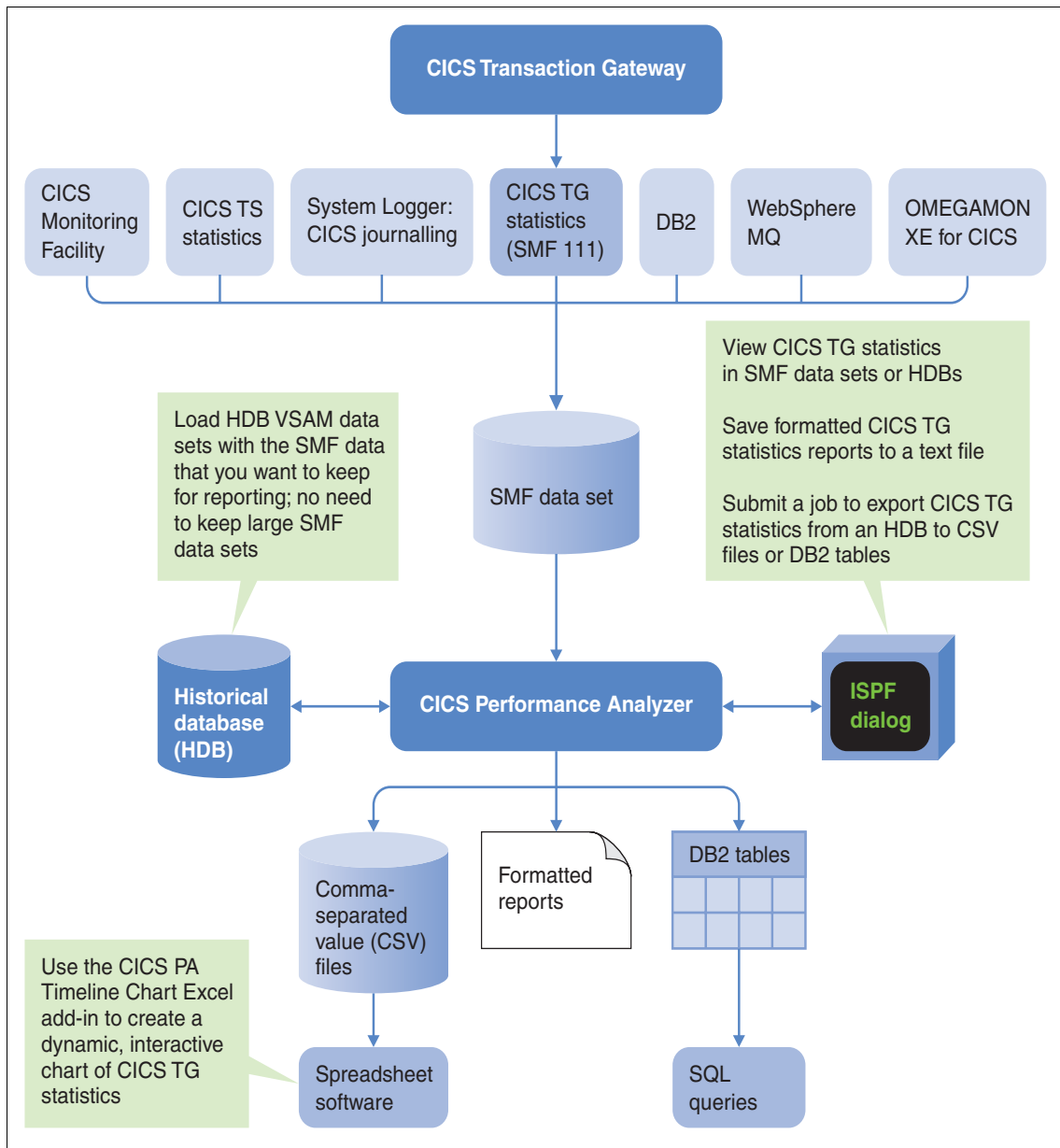


Figure 8-32 CICS PA support for CICS TG statistics (SMF 111) records

8.4.3 Viewing CICS TG statistics with CICS PA

To view CICS TG statistics in the CICS PA ISPF dialog, select the Statistics option from the primary menu. This displays the statistics reporting menu, as shown in Figure 8-33.

File Options Help			
CICS Statistics Reporting Menu			
Command ==> _____			
Select an option then press Enter.			
4	1. SMF Files defined in Personal System Definitions		
	2. SMF Files defined in Shared System Definitions		
	3. Historical Databases for CICS Statistics		
	4. Process SMF File		
	'PROD1.SMF.G0128V00' +		
Filter Criteria NO			
APPLID	_____	Start	_____ YYYY/MM/DD HH:MM:SS
Image	_____	Stop	_____
Type	_____ /	EOD	_____ / INT / USS / REQ / RRT
Options 2 and 3:			
HDB Register	_____		'CPA.HDB.REGISTER' +

Figure 8-33 CICS PA statistics reporting menu

From this menu, you can choose to view statistics in an SMF file or in an HDB. In either case, CICS PA displays a selection list screen of statistics intervals, as shown in Figure 8-34.

File Edit Filter Options Help									
REPORT				Statistics Intervals				Row 1 from 112	
Command ==>								Scroll ==> PAGE	
Select the required CICS Statistics interval.									
/	System	Image	VRM	Type	--- Collection Time ---		Reset	Duration	
s	CTG71AAA	MV23	710	TG INT	2008/03/11	21:14:00 Tue	00:05:00	00:05:00	
-	IYCNC TGS	MV23	650	TS INT	2008/03/11	21:14:00 Tue	21:09:00	00:05:00	
-	CTG71AAA	MV23	710	TG INT	2008/03/11	21:19:00 Tue	00:05:00	00:05:00	
-	IYCNC TGS	MV23	650	TS INT	2008/03/11	21:19:00 Tue	21:14:00	00:05:00	
-	CTG71AAA	MV23	710	TG INT	2008/03/11	21:24:00 Tue	00:05:00	00:05:00	
-	IYCNC TGS	MV23	650	TS INT	2008/03/11	21:24:00 Tue	21:19:00	00:05:00	
-	CTG71AAA	MV23	710	TG INT	2008/03/11	21:29:00 Tue	00:05:00	00:05:00	
-	IYCNC TGS	MV23	650	TS INT	2008/03/11	21:29:00 Tue	21:24:00	00:05:00	
-	CTG71AAA	MV23	710	TG INT	2008/03/11	21:34:00 Tue	00:05:00	00:05:00	
-	IYCNC TGS	MV23	650	TS INT	2008/03/11	21:34:00 Tue	21:29:00	00:05:00	
-	CTG71AAA	MV23	710	TG INT	2008/03/11	21:39:00 Tue	00:05:00	00:05:00	
-	IYCNC TGS	MV23	650	TS INT	2008/03/11	21:39:00 Tue	21:34:00	00:05:00	
-	CTG71AAA	MV23	710	TG INT	2008/03/11	21:44:00 Tue	00:05:00	00:05:00	
-	IYCNC TGS	MV23	650	TS INT	2008/03/11	21:44:00 Tue	21:39:00	00:05:00	
-	CTG71AAA	MV23	710	TG EOD	2008/03/11	21:47:11 Tue	00:03:11		
-	IYCNC TGS	MV23	650	TS USS	2008/03/11	21:47:14 Tue	21:44:00		
-	IYCNC TGS	MV23	650	TS INT	2008/03/11	21:49:00 Tue	21:44:00	00:05:00	
-	IYCNC TGS	MV23	650	TS INT	2008/03/11	18:54:00 Tue	18:49:00	00:05:00	
-	IYCNC TGS	MV23	650	TS INT	2008/03/11	18:59:00 Tue	18:54:00	00:05:00	
-	IYCNC TGS	MV23	650	TS INT	2008/03/11	19:04:00 Tue	18:59:00	00:05:00	
-	IYCNC TGS	MV23	650	TS INT	2008/03/11	19:09:00 Tue	19:04:00	00:05:00	
-	IYCNC TGS	MV23	650	TS INT	2008/03/11	19:14:00 Tue	19:09:00	00:05:00	
-	IYCNC TGS	MV23	650	TS INT	2008/03/11	19:19:00 Tue	19:14:00	00:05:00	
-	IYCNC TGS	MV23	650	TS INT	2008/03/11	19:24:00 Tue	19:19:00	00:05:00	

Figure 8-34 CICS PA statistics interval selection screen

The Type column indicates both the system type, such as TS for CICS Transaction Server or TG for CICS Transaction Gateway, and the collection type, such as INT or EOD. For a CICS TG system, the VRM column indicates the CICS TG VRM, such as 710 for V7.1.

Selecting a CICS TG interval displays the CICS TG statistics report tree, showing the CICS TG statistics reports, as shown in Figure 8-35 on page 339.

```
File Edit Options View Help
REPORT                               Statistics Reports                               Line 1 of 9
Command ==> _____ Scroll ==> CSR

System: CTG71AAA/MV23      Type: INT  Interval: 2008/03/11 21:14:00 Tuesday

--- - --- ** Reports **                      Size
      CICS Transaction Gateway                7
      S Connection Manager                    1
      CICS Server Statistics                  1
      CICS Server Instance for EXCI           0
      CICS Server Instance for IPIC           1
      Gateway Daemon                          1
      Protocol Handler                       1
      Worker Thread                          1
      System Environment                      1
      ** End of Reports **
```

Figure 8-35 CICS PA statistics reporting tree

These reports match the CICS TG statistics resource groups described in the CICS TG Information Center:

http://publib.boulder.ibm.com/infocenter/cicstg/v7r1m0/topic/com.ibm.ci.cs.tg.doc/ctgunx/stat_rg.html

Selecting a report displays the statistics data for the selected interval, as shown in Figure 8-36.

```
File Form Options Help
REPORT      Connection Manager                               Line 00000001
Command ==> _____ Scroll ==> CSR

System: CTG71AAA/MV23      Type: INT  Interval: 2008/03/11 21:14:00 Tuesday

Current Allocated Connection Managers . . : 100
Current Connection Managers . . . . . : 150
Current Connection Managers Waiting . . : 0
Lifetime ConnectTimeout Limit Hit . . . : 0
Initial Connection Managers . . . . . : 2
Max Connection Managers . . . . . : 200
Interval ConnectTimeout Limit Hit . . . : 0
Interval Peak Alloc Conn Mgr Threads . . : 100
Interval Conn Mgr Threads Created . . . : 0
Interval Alloc Conn Mgr Threads . . . . : 0
```

Figure 8-36 CICS PA displaying CICS TG Connection Manager statistics

Pressing function key F1 (Help) displays descriptions of each statistic, including a cross-reference to the CICS TG field names, as shown in Figure 8-37.

Field Descriptions for Statistics Report

Category : CICS Transaction Gateway	Macro . . : CTGSMFS1
Report . . : Connection Manager	DSECT . . : CTG_CM

More: +

Current Allocated Connection Managers

CICS field name: CTG_CM_CALLOC DB2 column name: C_ALLOC_CONN_MGRS

The current number of connection manager threads allocated to Java clients.

Reset characteristic: Reset to current value

Current Connection Managers

CICS field name: CTG_CM_CCURR DB2 column name: C_CONN_MGRS

The current number of connection manager threads created.

Reset characteristic: Reset to current value

Current Connection Managers Waiting

CICS field name: CTG_CM_CWAITING DB2 column name: C_CONN_MGRS_WAIT

The current number of connection managers waiting for a worker thread to become available.

Reset characteristic: Reset to current value

Figure 8-37 CICS PA online help for CICS TG statistics

Statistics reports that can have multiple rows are presented as tables, as shown in Figure 8-38.

File Edit Form Options Help						
CPAHSRPP CICS Server Instance for IPIC			Line 00000001 Col 002 007 >			
Command ==>			Scroll ==> CSR			
System: CTG71AAA/MV23 Type: INT Interval: 2008/03/11 21:14:00 Tuesday						
Server Name in CTG.INI	Interval Request Data Sent Bytes	Lifetime Request Data Sent Bytes	Interval Response Data Sent Bytes	Lifetime Response Data Sent Bytes	Interval IPIC Requests Processed	Lifet I Reque Proces
CTG0IPIC	13M	87M	6M	44M	26782	171

Figure 8-38 CICS PA displaying "CICS TG CICS Server Instance for IPIC statistics (tabular layout)

You can save these formatted reports to a text file: on the statistics reporting tree (Figure 8-35 on page 339), enter line action P (Print) next to one or more reports. CICS PA writes the formatted reports to the data set that you specify.

8.4.4 Exporting CICS TG statistics to DB2

You can use CICS PA HDB container data sets as input to the DB2 Load Utility to populate DB2 tables. CICS PA will generate a JCL that, in a single job, loads SMF 111 records from an SMF data set to an HDB, and then calls the DB2 Load Utility (DSNUTILB) to load those records from the HDB into DB2, as shown in Figure 8-39.

```
...
//DSNUPROC EXEC PGM=DSNUTILB, ...
//SYSIN DD *
LOAD DATA REPLACE
  INTO TABLE TEST.HSTG000A WHEN (70) = 'G00A' (
    START_DATE          POSITION(1)      DATE EXTERNAL(10),
    START_TIME          POSITION(12)     TIME EXTERNAL(8),
    APPLID              POSITION(20)     CHAR(8),
    MVSID               POSITION(28)     CHAR(4),
    VRM                 POSITION(44)     CHAR(3),
    INTERVAL_TYPE       POSITION(47)     CHAR(3),
    INTERVAL_DURATION   POSITION(50)     TIME EXTERNAL(8)
    NULLIF(50)=' ',
    INTERVAL_NUMBER     POSITION(58)     INTEGER,
    C_ALLOC_CONN_MGRS   POSITION(77)     INTEGER,
    C_CONN_MGRS         POSITION(81)     INTEGER,
    C_CONN_MGRS_WAIT    POSITION(85)     INTEGER,
  )
...
```

Figure 8-39 CICS PA uses the DB2 Load Utility to export CICS TG statistics from HDBs to DB2

CICS PA also generates a JCL for you that contain a DDL to create the DB2 tables, as shown in Figure 8-40.

```
...  
CREATE TABLE TEST.HSTG000A (  
    START_DATE          DATE,  
    START_TIME          TIME,  
    APPLID              CHAR(8),  
    MVSID               CHAR(4),  
    VRM                 CHAR(3),  
    INTERVAL_TYPE       CHAR(3),  
    INTERVAL_DURATION   TIME,  
    INTERVAL_NUMBER     INTEGER,  
    C_ALLOC_CONN_MGRS   INTEGER,  
    C_CONN_MGRS         INTEGER,  
    C_CONN_MGRS_WAIT    INTEGER,  
    L_TIMEOUTS_LIMIT    INTEGER,  
    INIT_CONN_MGRS      INTEGER,  
    MAX_CONN_MGRS       INTEGER,  
    I_TIMEOUTS_LIMIT    INTEGER,  
    I_PEAK_CONN_MGRS    INTEGER,  
    I_CREATE_CONN_MGRS  INTEGER,  
    I_ALLOC_CONN_MGRS   INTEGER  
    ) IN TEST.REDBOOK;  
...
```

Figure 8-40 CICS PA generates DDL to create DB2 tables for CICS TG statistics

For a detailed cross-reference between CICS TG statistics field names and the DB2 column names in DB2 tables generated by CICS PA, see the CICS PA SupportPac CP12.

8.4.5 Exporting CICS TG statistics to spreadsheet applications

You can export CICS TG statistics from a CICS PA statistics HDB to comma-separated value (CSV) files for use in PC-based spreadsheet applications, as shown in Figure 8-41 on page 343.

```
Start
Time,APPLID,MVSID,VRM,Interval_Type,Interval_Duration,Interval_Number,Current_Alloc
ated_Connection_Managers,...
2007-10-24-09.20.00,SCSCT717,SC66      ,710,INT,00:05:00,0,0,500,0,0,500,500,0,0,0,0
...
```

Figure 8-41 CSV file of CICS TG statistics (SMF 111 records) exported from CICS PA

8.4.6 Charting CICS TG statistics

You can transfer the CSV files generated by CICS PA from z/OS to your PC, and then use them with spreadsheet software, such as Microsoft Office Excel, to create charts and perform further analysis.

Example charts

Figure 8-42 on page 344 through Figure 8-47 on page 349 show example charts of CICS TG statistics data in CSV files generated by CICS PA.

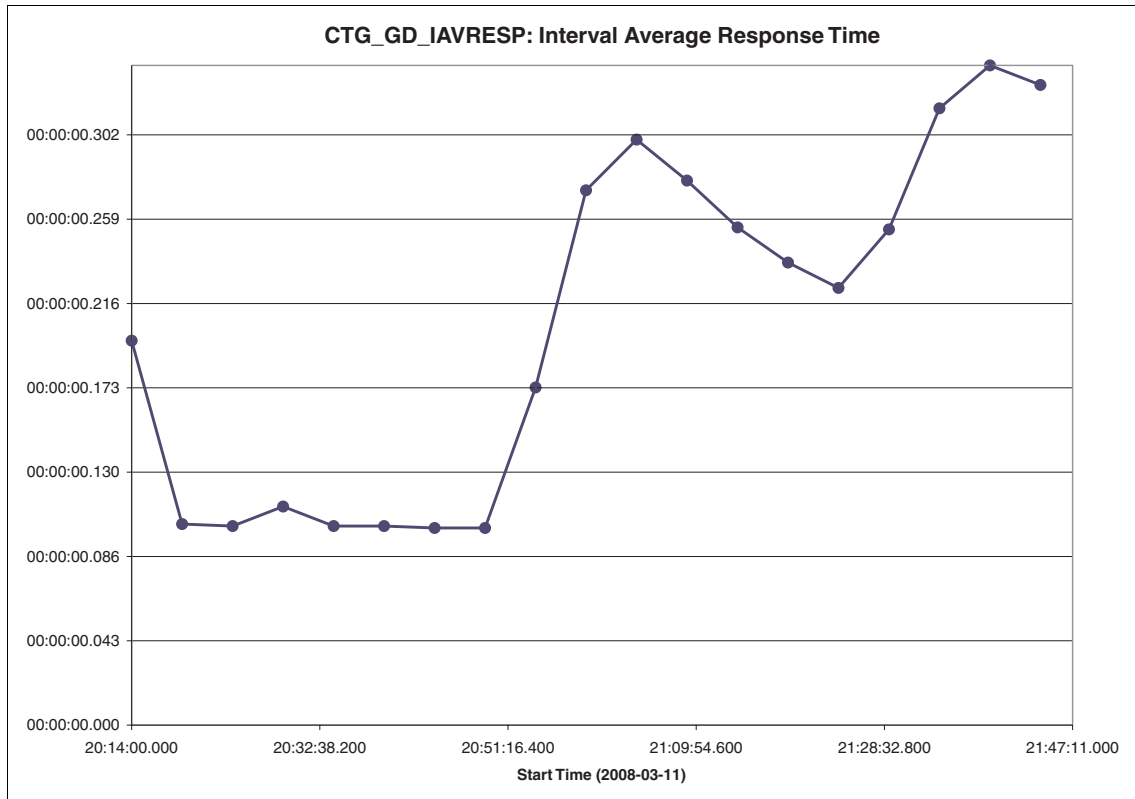


Figure 8-42 Charting CICS TG statistics: CTG_GD_IAVRESP

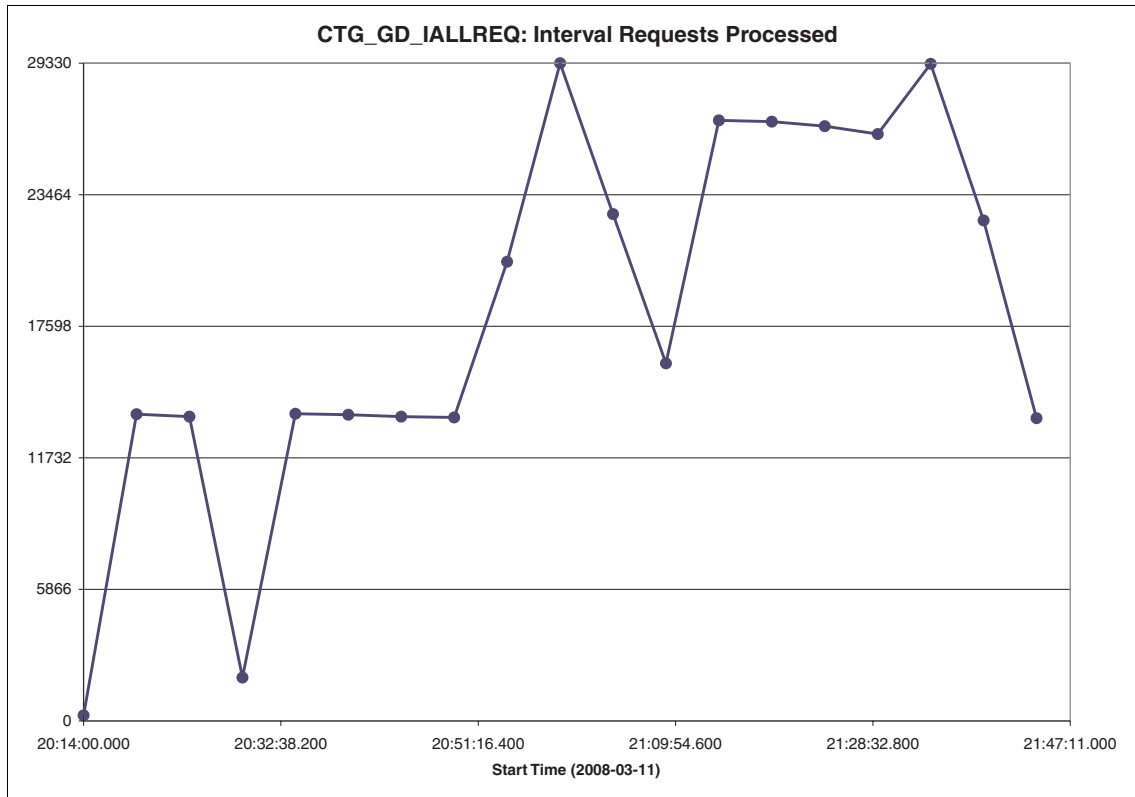


Figure 8-43 Charting CICS TG statistics: CTG_GD_IALLREQ

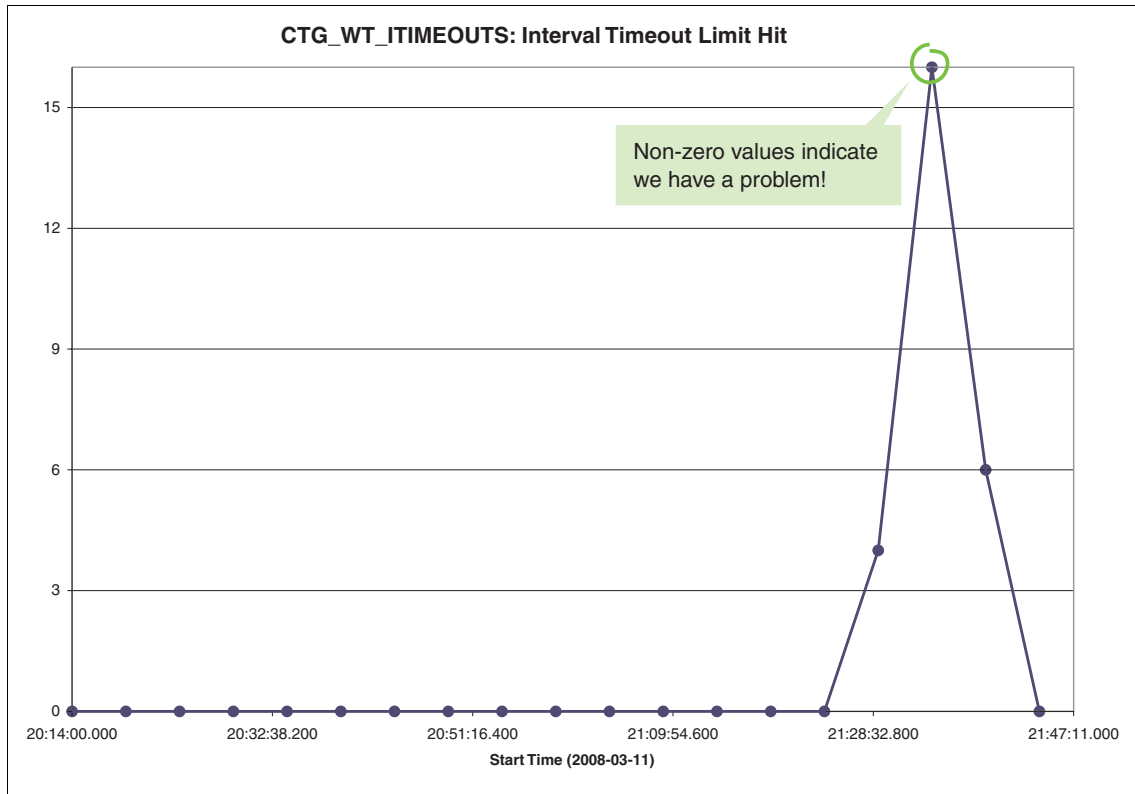


Figure 8-44 Charting CICS TG statistics: CTG_WT_ITIMEOUTS

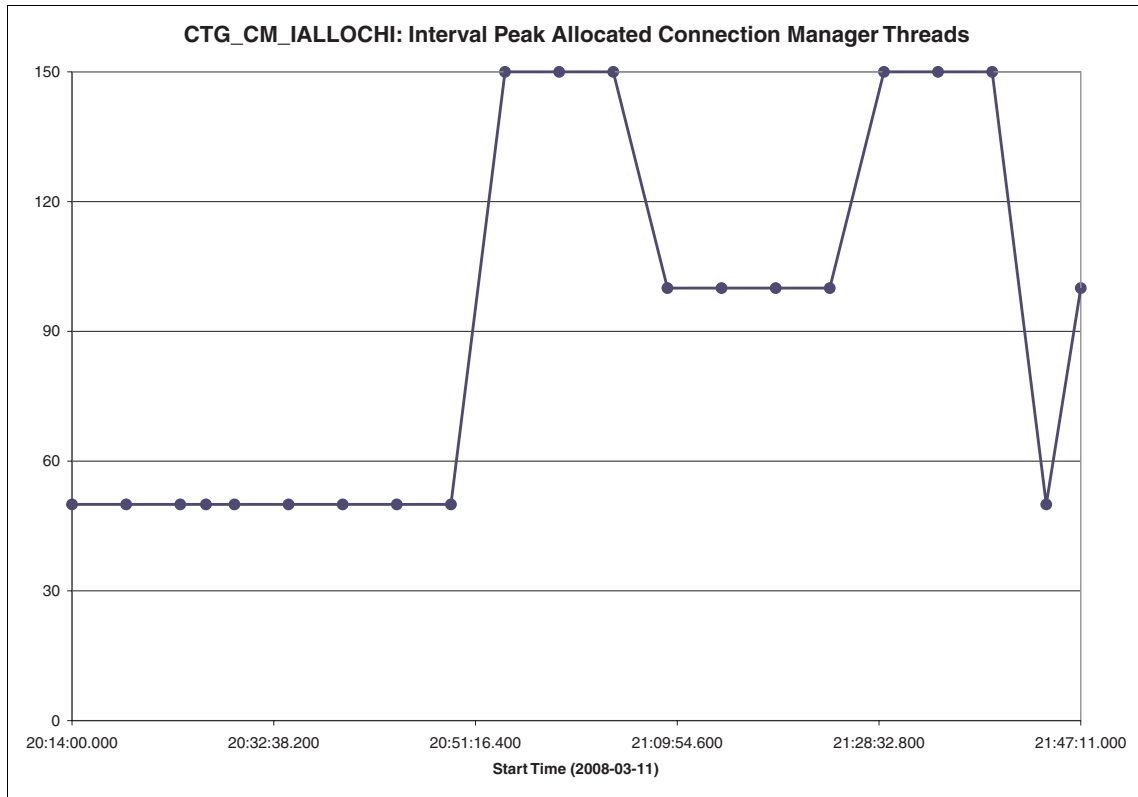


Figure 8-45 Charting CICS TG statistics: CTG_CM_IALLOCHI

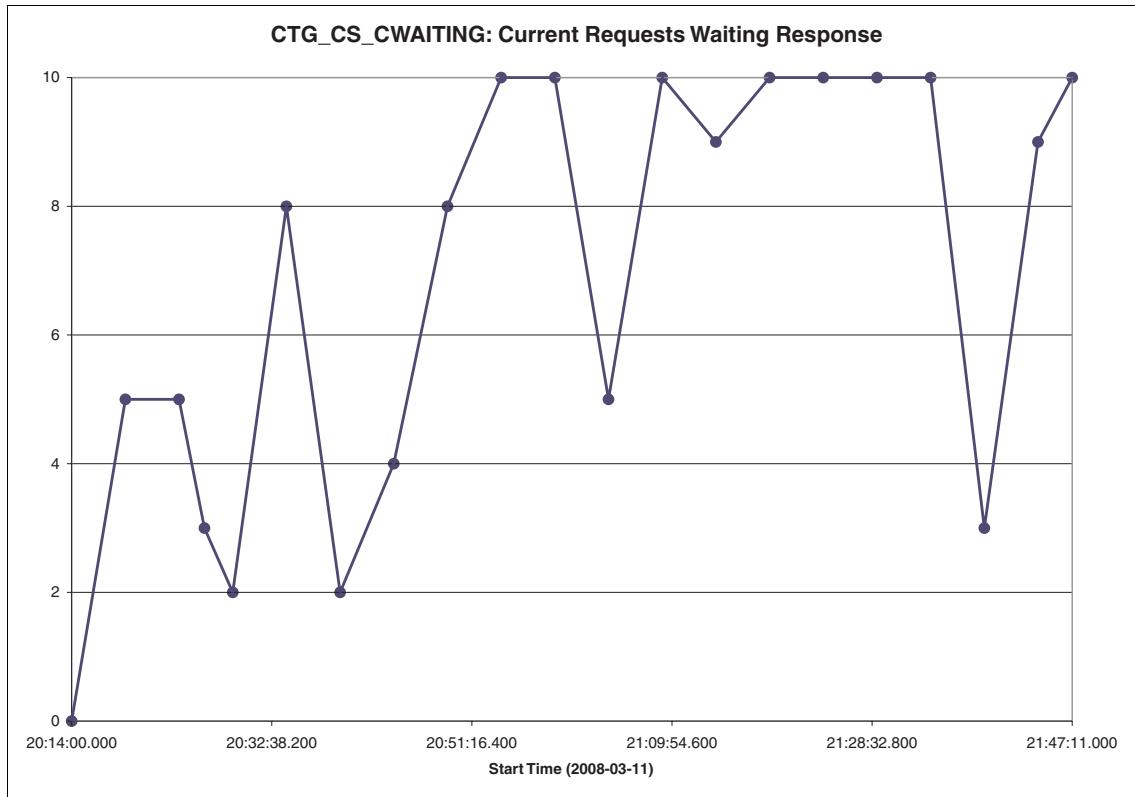


Figure 8-46 Charting CICS TG statistics: CTG_CS_CWAITING

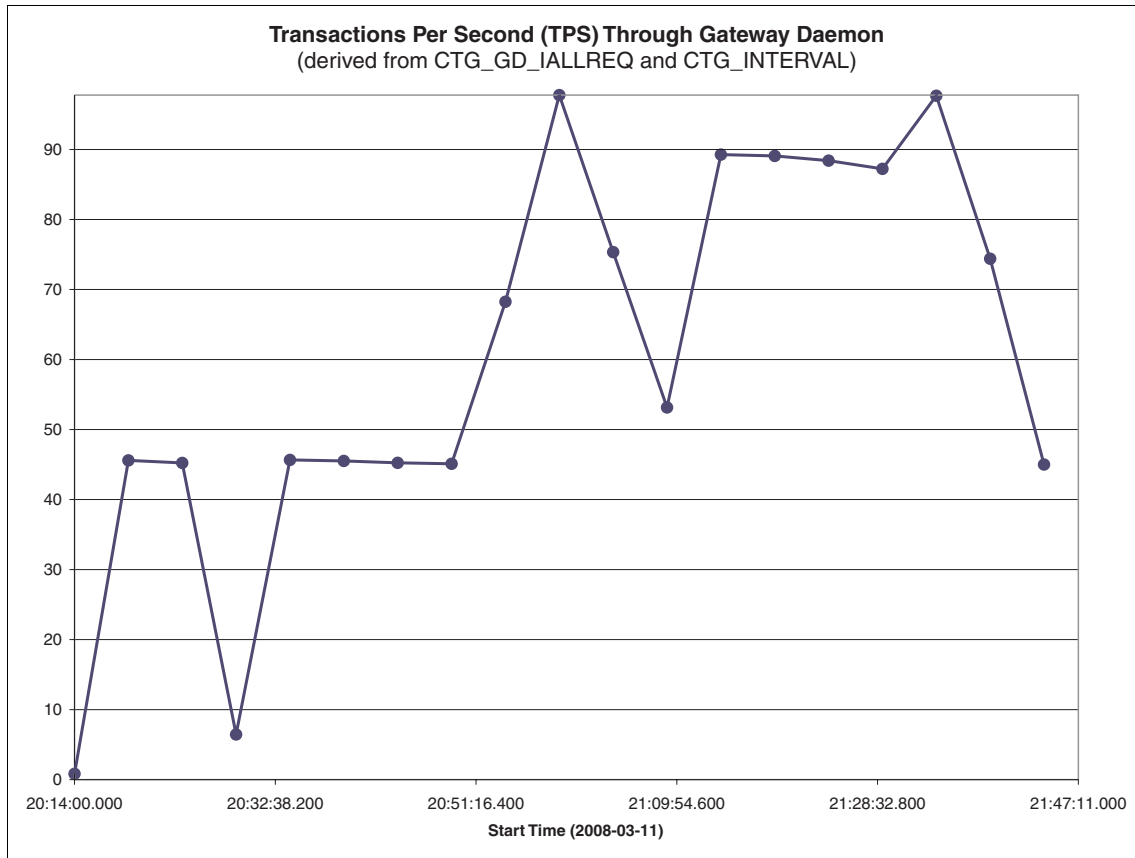


Figure 8-47 Charting CICS TG statistics: transactions per second (TPS) through Gateway Daemon

To create the chart in Figure 8-47, we used Excel to add a new column to the CICS PA-generated data, and then we defined the values for the new column using the following formula:

`= 'Interval Requests Processed' / ((HOUR('Interval Duration') * 3600) + (MINUTE('Interval Duration') * 60) + SECOND('Interval Duration'))`

This formula divides the number of requests processed by the Gateway Daemon per interval by the number of seconds in an interval duration (hence the parts of the formula that convert the hour and minute portions of the interval duration into seconds). Interval Requests Processed and Interval Duration refer to the column headings in the CSV file generated by CICS PA; these column headings correspond to the CICS TG field names CTG_GD_IALLREQ and CTG_INTERVAL.

CICS PA Timeline Chart Excel add-in

CICS PA SupportPac CP12 includes a CICS PA Timeline Chart Excel add-in that creates dynamic, interactive timeline charts of CICS PA-generated CSV files, as shown in Figure 8-48.

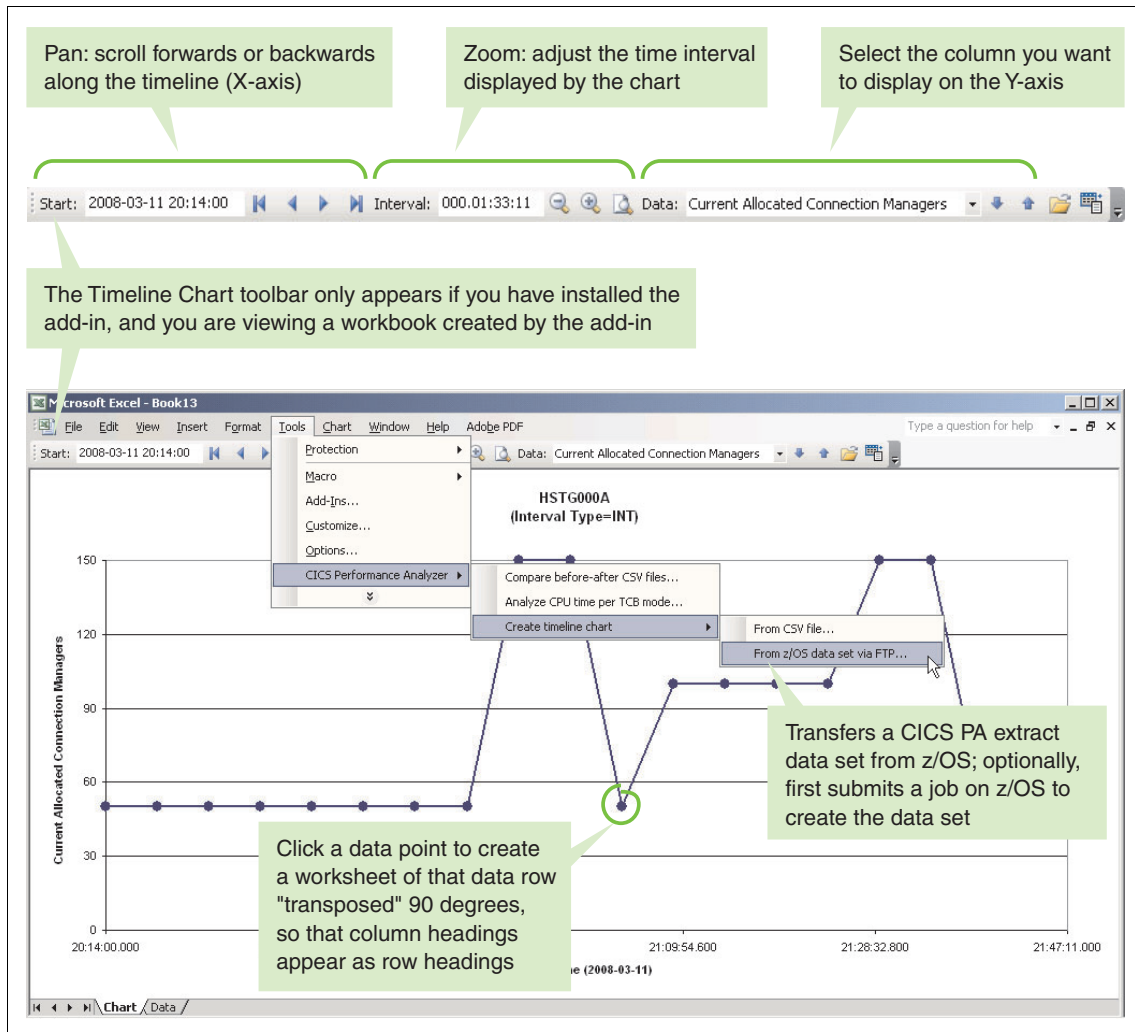


Figure 8-48 Interactive chart of CICS TG statistics created by CICS PA Timeline Chart Excel add-in (supplied with CICS PA SupportPac CP12)

You can download the SupportPac from the Web at:

<http://www.ibm.com/support/docview.wss?uid=swg24011321>



Part 3

Appendixes



Sample J2EE application - CTGTesterCCIXA

In this appendix, we provide details about our sample J2EE application, CTGTesterCCIXA.

The CTGTesterCCIXA application is intended to be used with the CICS ECI XA resource adapter. It contains two JCA resource references and can be used to flow two ECI requests to different CICS regions, so that the two calls can be committed within the same global two-phase commit (2pc) transaction.

Note that this application is designed for the testing of connectivity from the application server to CICS, either with a local-mode CICS TG or through the Gateway daemon (remote-mode). As such, all the CICS dependencies, such as COMMAREA size, input, encoding, and length, are externalized to enable you to test different configurations. The applications are not designed for production use, because such externals are unlikely to be exposed in real-life customer applications.

The CTGTesterCCIXA application

The CTGTesterCCIXA application provides a simple enterprise application that can be used *out-of-the-box* to test the XA capabilities of the CICS TG ECI XA resource adapter. It can be used to test connections to two different CICS regions (Figure A-1). The pair of JCA requests can form two legs of an XA transaction. This application was used extensively in Chapter 7, “High availability with XA and OMEGAMON XE” on page 241, and instructions on how to download our sample code are provided in Appendix D, “Additional material” on page 379.

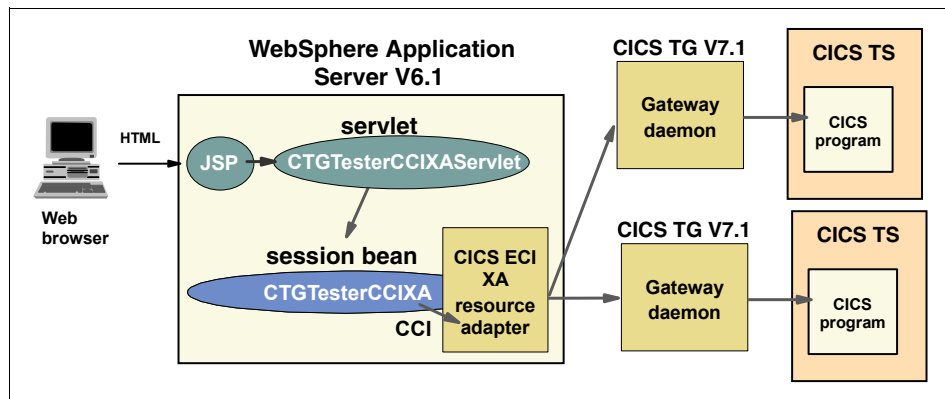


Figure A-1 Architecture of CTGTesterCCIXA

Application overview

CTGTesterCCIXA contains two JCA resource references. When deploying the application, the resource references must be mapped to two different connection factories.

The following is the sequence of events that occur when the user interacts with the CTGTesterCCIXA application through a Web browser.

1. The user opens the Web application using the following URL and enters the input parameters for the two JCA requests:

`http://9.12.4.111:9080/CTGTesterCCIXAWeb/`

2. The user then clicks a button on a Web page that submits a form to the servlet, as shown in Figure A-2 on page 355.

Modified CTGTesterCCIXA Start - Microsoft Internet Explorer

Address http://9.12.4.111:9080/CTGTesterCCIXAWeb/

Redbooks

CTGTesterCCIXA

version SG24-7562

This J2EE application, uses the JCA to call two COMMAREA based CICS programs in managed mode.

Transaction leg 1

CICS program name: Program on CICS to call

COMMAREA

Input data:

Length:

Codepage: (for example ASCII or IBM037)

Options

Usrid:

Password:

Mirror transaction:

Iterations: Number of times to run the program

Request delay: Request delay after transaction leg 1

Transaction leg 2

CICS program name: Program on CICS to call

COMMAREA

Input data:

Length:

Codepage: (for example ASCII or IBM037)

Options

Usrid:

Password:

Mirror transaction:

Iterations: Number of times to run the program

Request delay: Request delay after transaction leg 2

Common options

Application trace: T class trace

Request rollback: Request transaction rollback after transaction leg 2

Figure A-2 CTGTesterCCIXA index page

3. The servlet receives the request for action and calls a method on the remote interface of the CTGTesterCCIXA session bean.
4. The session bean uses the CICS ECI XA resource adapter to call two CICS programs, potentially through two different Gateway daemons.
5. The CICS ECI XA resource adapter returns data from the CICS programs to the session bean.
6. The session bean returns the output data from the CICS programs back to the servlet.

- The servlet forwards to a JSP™, which displays the response to the user, as shown in Figure A-3.

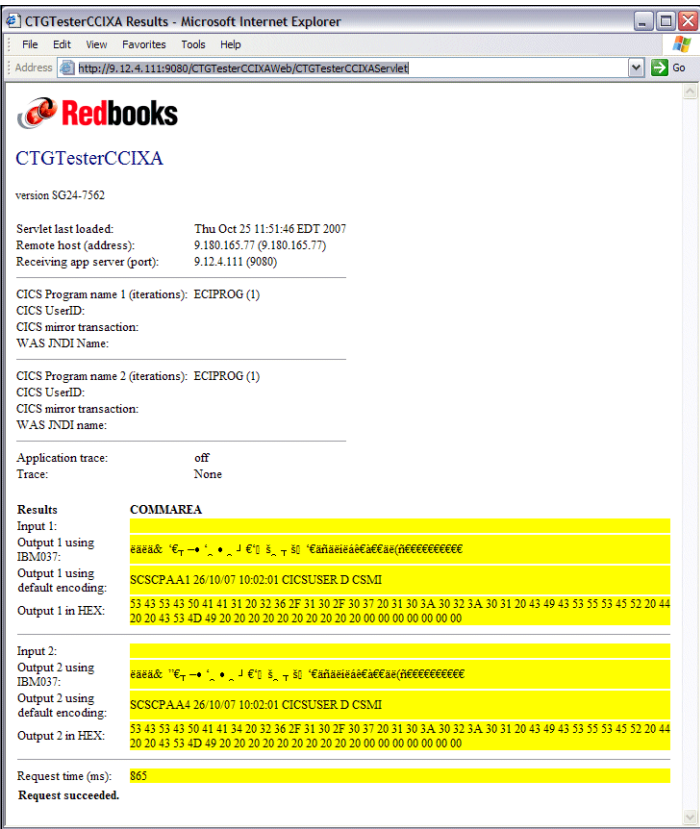


Figure A-3 CTGTesterCCIXA results page

Static HTML Form

The HTML form index.jsp is stored in the WebContent folder of the CTGTesterCCIXAWeb package and can be viewed using Rational® Application Developer. The HTML form contains the same fields used in CTGTesterCCIWeb, but includes a numeric suffix to distinguish the sets of parameters. Table A-1 shows the fields of the form.

Table A-1 Fields of the HTML form

Field	Name	Purpose
Mandatory details		
CICS program name	funcName{1+2}	Program on CICS to call
COMMAREA input	commareaInput{1+2}	COMMAREA data

Field	Name	Purpose
COMMAREA length	commareaLength{1+2}	Length of the COMMAREA
Encoding	encoding{1+2}	Encoding to convert data to before sending and after receiving
Optional details		
User ID	username{1+2}	User name to flow on the ECI request
Password	password{1+2}	Password to flow on the ECI request
Mirror transaction	mirror{1+2}	Transaction to use on the CICS server
Iterations	iterations{1+2}	The number of times to run the CICS program
Application trace	appTrace (common field)	Whether to enable CICS TG Java client trace

The action of the HTML form is to call the servlet CTGTesterCCIXAServlet, using the POST method to send the parameters.

CTGTesterCCIXAServlet

The Servlet converts the HTML FORM user data into a format that the Session EJB can use. The Session EJB facade masks how or where the request is actually processed, providing separation between presentation and business logic. Figure A-4 summarizes the CTGTesterCCIXAServlet code.

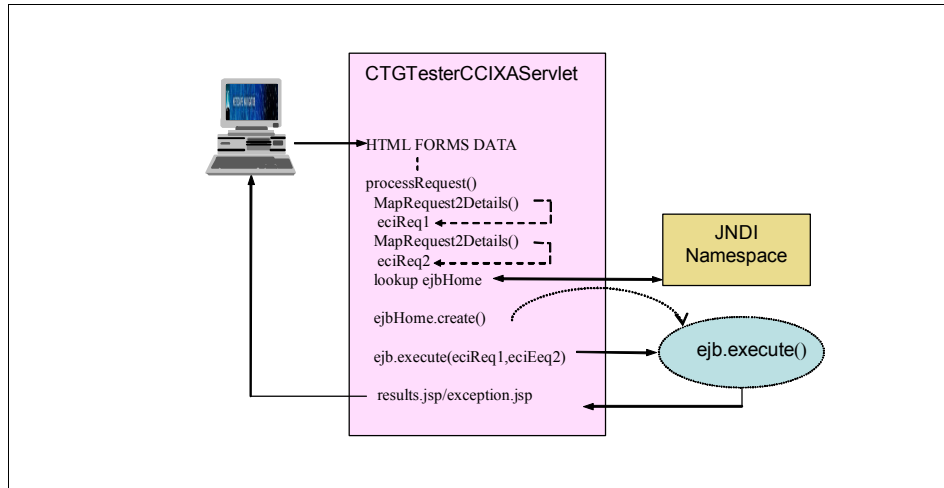


Figure A-4 CTGTesterCCIXAServlet logic overview

The following gives a brief description of the main methods of CTGTesterCCIXAServlet:

► processRequest() - 1

The CTGTesterCCIXAServlet doGet() and doPost() methods both pass on the HTML form data for the request (whichever convention was used) to the processRequest() method. processRequest() creates a pair of ECIRequestDetails objects eciReq1 and eciReq2, and calls MapRequest2Details() twice to populate them, using the suffix parameter to distinguish the sets of name/value pairs.

► MapRequest2Details()

The MapRequest2Details() method takes a HttpServletRequest object (effectively the input) and an ECIRequestDetails object (effectively the output) together with an optional suffix string. The field names, shown in the central column of Table A-1 on page 356, are combined with the suffix input string to identify name/value pairs contained in the HttpServletRequest data. The value for each field is copied into the corresponding EciRequestDetails class variable, performing any type conversions from the user input that are required.

MapRequest2Details() is invoked with the suffix 1 to populate eciReq1, and then again with suffix 2 to populate eciReq2.

► processRequest() - 2

When the request data has been set up in objects eciReq1 and eciReq2, processRequest() obtains a reference to CTGTesterCCIHome through a lookup on the initial context for EJB reference ejb/CTGTesterCCIXA. The CTGTesterCCIXAHome.create() method is used to create an instance of the CTGTesterCCIXA EJB remote interface. The local CTGTesterCCIXA instance is called tester. The ejb reference is defined in the Web Deployment Descriptor.

Having gathered together all of the details for the pair of JCA requests and created an instance of the Session EJB CTGTesterCCIXA remote interface, processRequest() finally invokes the execute() method on the CTGTesterCCIXA bean instance, passing in both EciRequestDetails objects, eciReq1 and eciReq2.

Session EJB CTGTesterCCIXA

Figure A-5 shows the interactions between the calling servlet and session EJB CTGTesterCCIXA.

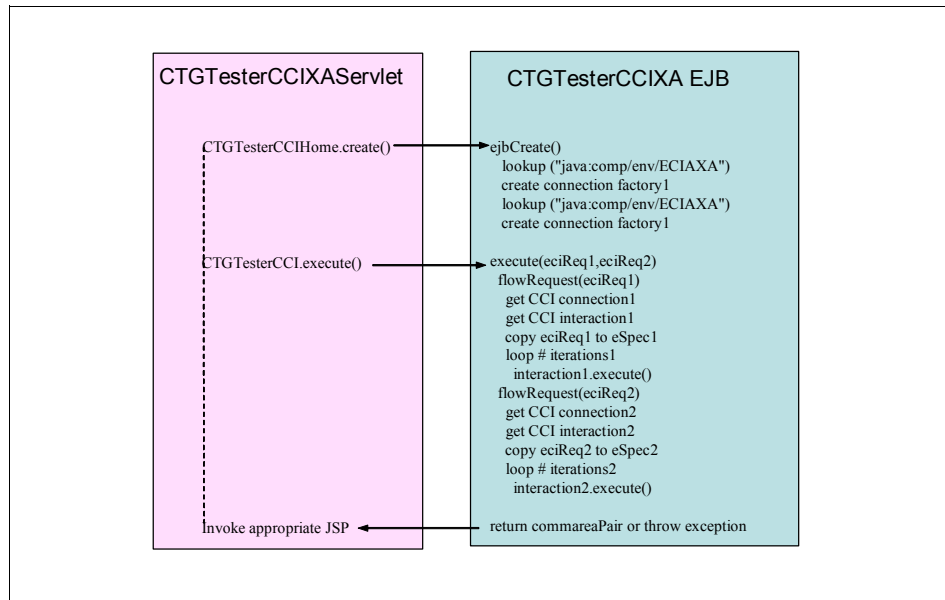


Figure A-5 CTGTesterCCIXA EJB logic overview

We give a brief description of the main methods of CTGTesterCCIXA in the following sections.

ejbCreate()

The EJB life cycle method `ejbCreate()` is used to create two connection factories. We create an `InitialContext` object and use it to look up the two connection factories using the JNDI names `java:comp/env/ECIAXA` and `java:comp/env/ECIBXA`. These are the local JNDI names that will be mapped to the actual JNDI name of the connection factories at deploy time. To indicate that we want such a mapping, we create two resource references in the EJB deployment descriptor with the names `ECIAXA` and `ECIBXA` (Figure A-6).

```
<resource-ref id="ResourceRef_1126776812792">
  <description>CICS XA ECI Resource adapter to CICS region A</description>
  <res-ref-name>ECIAXA</res-ref-name>
  <res-type>javax.resource.cci.ConnectionFactory</res-type>
  <res-auth>Container</res-auth>
  <res-sharing-scope>Shareable</res-sharing-scope>
</resource-ref>
<resource-ref id="ResourceRef_1126872415486">
  <res-ref-name>ECIBXA</res-ref-name>
  <res-type>javax.resource.cci.ConnectionFactory</res-type>
  <res-auth>Container</res-auth>
  <res-sharing-scope>Shareable</res-sharing-scope>
</resource-ref>
```

Figure A-6 CTGTesterCCIXA EJB Deployment Descriptor

execute()

The only method available on the remote interface of the `CTGTesterCCIXA` EJB is the `execute()` method. This method and its supporting logic is implemented by the class `CTGTesterCCIXABean`. The `EciRequestDetails`, `commareaPair`, and `JavaStringRecord` classes also provide supporting functions.

Note: The transaction attribute is set in the assembly descriptor section of the EJB deployment descriptor for the `execute()` method to control the transactional behavior of the EJB and associated JCA requests. In the version of `CTGTesterCCIXA` provided with this book, the transaction attribute is set to *Required*.

The `execute()` method calls `flowEciRequest()` twice:

- ▶ The first request processes `eciReq1`, using `ECIInteractionSpec eSpec1` and the resource reference `ECIAXA`.
- ▶ The second request processes `eciReq2`, using `ECIInteractionSpec eSpec2` and the resource reference `ECIBXA`.

The CTGTesterCCIXA.execute() method then returns the two COMMAREAS in a commareaPair object, which consists of a pair of String objects.

Note: The transaction attribute is set in the assembly descriptor section of the EJB deployment descriptor for the execute() method to control the transactional behavior of the EJB and associated JCA request. In the version of CTGTesterCCIXA provided with this book, the transaction attribute is set to *Required*.

flowEciRequest()

The flowEciRequest() method creates a connection by calling the getConnection() method. The getConnection() method creates an EciConnectionSpec and sets security details if specified on the JSP input page.

Method flowEciRequest() then creates an interaction by calling getInteraction() and sets up the EciInteractionSpec object (eSpec) based on the other input parameters entered on the JSP input page (CICS program, COMMAREA length, and mirror transaction name). Finally, it enters a loop based on the iteration limit entered on the JSP input page, issuing the execute() method on the interaction (eciInt).

The execute() method takes the JavaRecordString object jsr as a parameter twice, once for COMMAREA input and once for COMMAREA output.

The iteration loop continues for as long as each request completes normally and until the iteration limit is reached, each time overwriting the output COMMAREA of the previous iteration.

Ultimately, the invocation of flowEciRequest() either returns a JavaStringRecord, representing the COMMAREA returned by the last iteration (if successful) or re-throws a ResourceException caught on the EciInteraction.execute() call.

Catching and handling a CICS transaction abend

The execute() method might throw a ResourceException, which has several sub-classes. For example, in the event of a CICS transaction abend, a CICSTxnAbendException is thrown by the CICS ECI resource adapter. If this exception occurs, we issue a setRollbackOnly() on the EJB session context in the catch block for the execute() call. This ensures that the CICS abend will be propagated onto WebSphere Application Server as the transaction manager and the local transaction (if one exists) will be rolled back. Example A-1 shows this logic.

Example: A-1 Catching and handling a transaction abend

```
public class CTGTesterCCIXABean implements SessionBean {
    ...
    private JavaStringRecord flowEciRequest(ECIInteractionSpec eSpec,
        EciRequestDetails eciRD, String cfRef) throws Exception {

    ...
        // make the call
        try {
            eciInt.execute(eSpec, jsr, jsr);
        } catch (ResourceException re) {
            // output the stacktrace and re-throw it back to the client
            re.printStackTrace();
            // Depending upon the kind of exception thrown, we may decide
            // to abandon the current transaction by calling setRollbackOnly()
            // on the current context. If this request is not part of a local
            // or global transaction, there will be a second exception to
            // catch (IllegalStateException). If so, we will log the exception
            // details, but otherwise absorb it.

            // In the case of a CICSTxnAbendException, we will write a log
            // message and rollback any current transaction.
            if (re instanceof com.ibm.connector2.cics.CICSTxnAbendException )
            {
                System.err.println("CTGTesterCCIXABean.flowEciRequest: CICS ABEND
detected."+
                    " Connection Factory="+targetCF.toString());
                try {
                    mySessionCtx.setRollbackOnly();
                } catch (IllegalStateException ise) {
                    ise.printStackTrace();
                }
            } else {
                // Write generic failure log message
                System.err.println("CTGTesterCCIXABean.flowEciRequest:
ResourceException detected."+
                    " Connection Factory="+targetCF.toString());
            }

            // clean up interaction and connection references
            dropConnection(eciConn, eciInt);
        }
    }
}
```

```

        eciInt = null;
        eciConn = null;
        // throw exception back to caller
        ResourceException re2 = new ResourceException(re.getMessage()+
            " ConnectionFactory="+targetCF.toString());
        re2.initCause(re);
        throw re2;
    }
}

```

JSPs

Note: The call to the `setRollbackOnly` method shown in Example A-1 is not necessary if the option `ABENDBKOUT=YES` is set in the `DFHXCOPT` table. For further details refer to the [ibm.com](http://www.ibm.com/support/docview.wss?uid=swg21248201) support article found at:

<http://www.ibm.com/support/docview.wss?uid=swg21248201>

The application uses two JSPs to format and display the results. One of two scenarios can occur: The request completes successfully, or an exception occurs inside the CICS ECI XA resource adapter or the application. The JSPs are stored in the `CTGTesterCCIXAWeb` WebContent folder and can be viewed using Rational Application Developer.



B

SMF Historical Data

This appendix provides a statistics report of SMF type111 records produced by the sample CICS TG record viewer program CTGSMFRD. The report was used in Chapter 8, “Historical data analysis” on page 293. The following sample contains SMF interval statistics as reported by the supplied sample program (see Example B-1 on page 366).

SMF interval statistics report

Details of the SMF fields and their format can be found in the section *CICS Transaction Gateway SMF record* on the CICS TG InfoCenter.

Example: B-1 SMF interval statistics displayed using the sample CTGSMFRD

CICS TG SMF dataset viewer.

-----START OF RECORD-----

Header section

SMF111_LEN (Record length) = 714

SMF111_SPN (APPLID) = SCST711

SMF111_SID (Subsystem identifier) = SC66

SMF111_VRM (CICS TG version) = 710

CTG_STATTYPE= INTERVAL

CTG_LOCOFFSET (Offset in secs from GMT) = -18000

CTG_COLTIME (collection time) = 14:00:00

CTG_COLDATE year = 2007, days from start of year = 288

CTG_INTVCOUNT (interval count) = 1

Data section

Resource Group: type = CM

CTG_CM_CALLOC = 400

CTG_CM_CCURR = 500

CTG_CM_CWAITING = 146

CTG_CM_LTIMEOUTS = 0

CTG_CM_SINIT = 500

CTG_CM_SMAX = 500

CTG_CM_ETIMEOUTS = 0

CTG_CM_IALLOCCHI = 400

CTG_CM_ICREATED = 0

CTG_CM_IALLOC = 0

0Resource Group: type = CS

CTG_CS_CALLOC = 100

CTG_CS_LALLOCFAIL = 0

CTG_CS_LALLREQ = 518278

CTG_CS_LCOMMSFAIL = 0

CTG_CS_LCOUNT = 1

CTG_CS_LREALLOC = 0

CTG_CS_SLOGONLIM = 100

CTG_CS_NETNAME = SCST711

CTG_CS_IALLOCFAIL = 0

CTG_CS_IALLREQ = 518298

CTG_CS_ICOMMSFAIL = 0

CTG_CS_ICOUNT = 1

CTG_CS_IREALLOC = 0

```

CTG_CS_IREQDATA = 51834600
CTG_CS_LREQDATA = 51834600
CTG_CS_IRESPPDATA = 51827800
CTG_CS_LRESPPDATA = 51827800
CTG_CS_SCOUNT = 0
CTG_CS_ICONNFFAIL = 0
CTG_CS_LCONNFAIL = 0
CTG_CS_ILOSTCONN = 0
CTG_CS_LLOSTCONN = 0
CTG_CS_LIDLETIMEOUT = 0
CTG_CS_CSESSCURR = 0
CTG_CS_CSESSMAX = 0
CTG_CS_LSESSFFAIL = 0
CTG_CS_ISESSFFAIL = 0
CTG_CS_CWAITING = 68
CTG_CS_IIDLETIMEOUT = 0
CTG_CS_IAVRESP = 513
CTG_CS_LAVRESP = 513
0Resource Group: type = WT
CTG_WT_CALLOC = 97
CTG_WT_CCURR = 100
CTG_WT_LTIMEOUTS = 3406
CTG_WT_SINIT = 100
CTG_WT_SMAX = 100
CTG_WT_ETIMEOUTS = 3406
CTG_WT_IALLOCHI = 100
0Resource Group: type = PH
CTG_PH_SPORT_SSL = -1
CTG_PH_SPORT_TCP = 2006
0Resource Group: type = SE
CTG_SE_SELIM = 524288000
CTG_SE_CELOAL = 443908096
CTG_SE_CHEAPGCMIN = 8320848
CTG_SE_SHEAPINIT = 134217728
CTG_SE_SHEAPMAX = 134217728
CTG_SE_IGCTIME = 299
CTG_SE_LGCTIME = 299
CTG_SE_IGCCOUNT = 13
CTG_SE_LGCCOUNT = 13
0Resource Group: type = GD
CTG_GD_HEALTH = 100
CTG_GD_STATUS = RUNNING
CTG_GD_LALLREQ = 518298
CTG_GD_LLUWTXNC = 0
CTG_GD_LLUWTXNR = 0

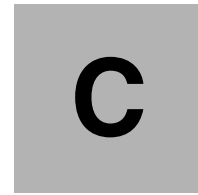
```

```

CTG_GD_LRUNTIME = 2941
CTG_GD_LSYNCTXN = 518325
CTG_GD_LXATXNC = 0
CTG_GD_LXATXNR = 0
CTG_GD_SNAME = SC SCT711
CTG_GD_IALLREQ = 518298
CTG_GD_IRUNTIME = 2941
CTG_GD_ISYNCTXN = 518325
CTG_GD_IXATXNC = 0
CTG_GD_IXATXNR = 0
CTG_GD_ILUWTXNC = 0
CTG_GD_ILUWTXNR = 0
CTG_GD_IAVRESP = 991
CTG_GD_LAVRESP = 991
CTG_GD_IREQDATA = 113809642
CTG_GD_LREQDATA = 113809642
CTG_GD_IRESPDATA = 53986217
CTG_GD_LRESPDATA = 53986217
CTG_GD_CSYNCTXN = 70
CTG_GD_CLUWTXN = 0
CTG_GD_CXATXN = 0
CTG_GD_LXAREQ = 0
CTG_GD_IXAREQ = 0
CTG_GD_SAPPLID =
CTG_GD_SAPPLIDQ =
CTG_GD_SSTATINT = 010000
CTG_GD_SSTATEEOD = 000000
CTG_GD_CNEXTRRESET = 140000
0Resource Group: type = CSX_EXCI
CTG_CSX_SAPPLID = SC SCPAA1
CTG_CSX_CALLOC = 100
CTG_CSX_LALLOCFAIL = 100
CTG_CSX_LALLREQ = 518325
CTG_CSX_IALLOCFAIL = 0
CTG_CSX_IALLREQ = 518325
CTG_CSX_IAVRESP = 513
CTG_CSX_LAVRESP = 513
CTG_CSX_LCOMMSFAIL = 0
CTG_CSX_IREQDATA = 51840200
CTG_CSX_LREQDATA = 51840400
CTG_CSX_IRESPDATA = 51832900
CTG_CSX_LRESPDATA = 51832900
CTG_CSX_SPROTOCOL = EXCI
CTG_CSX_ICOMMSFAIL = 0
CTG_CSX_CWAITING = 85

```

0-----END OF RECORD-----



Sample JCL for SMF reports

This appendix provides the sample JCL used in Chapter 8, “Historical data analysis” on page 293 for the production of SMF reports and the link-edit of the CICS TG record viewer sample program CTGSMFRD.

JCL to link-edit CTGSMFRD

Example C-1 contains the JCL we used to link-edit the CICS TG record viewer sample program CTGSMFRD (see “Accessing the CICS TG SMF records” on page 296).

This JCL was based on sample job CTGSMFB found in the CICS TG SCTGSAMP.

Example: C-1 JCL used to link-edit CTGSMFRD

```
//CTGSMFB JOB (0),MSGCLASS=X,CLASS=A,NOTIFY=&SYSUID,REGION=96M
/*
/* Sample JCL to compile and linkedit the SMF record viewer
/* program CTGSMFRD.
/*
/* @start_copyright_sample_ZOS@
/* Licensed Materials - Property of IBM
/*
/* 5655-R25
/*
/* (c) Copyright IBM Corp. 2007
/*
/* US Government Users Restricted Rights - Use, duplication or
/* disclosure restricted by GSA ADP Schedule Contract with
/* IBM Corp.
/*
/* Status: Version 7 Release 1
/* @end_copyright_sample_ZOS@
/*
/* Before running this JCL, you will need to modify the JOB
/* control card and globally replace the installation dependent
/* variables:
/*   <Install hlq>
/*       The prefix set at install time for a specific CICS TG
/*       version of load libraries. e.g. CICSTG.CTG@PRODUCT_VRM_NO_SEP@
/*   <EXE PDSE>
/*       The user data set to receive the text deck from the
/*       compiler step. Suggested PDS properties:
/*       ORG=PO,RECFM=U,RECLEN=0,BLOCKSIZE=6144,TYPE=LIBRARY
/*   <SC EE PREFIX>
/*       The dataset prefix for the SC EE* libraries, specific
/*       to the version of z/OS being used.
/*   <SCCN CMP PREFIX>
/*       The dataset prefix for the SCCN CMP library, specific
```



```

/**      to the version of z/OS being used.
/**
/** This job will compile and linkedit the CICS TG SMF viewer
/** sample program CTGSMFRD.
/**
/** Note that the program will use XPLINK style linkage and
/** LE runopt POSIX(ON).
/**
/** Definition step: create variables for rest of JCL
/**
/** CTGHLQ is the dataset prefix used by this CICS TG installation.
/** LTARG is a pre-allocated PDSE to receive the executable
/** output from the binder step.
/** LIBPRFX is the dataset prefix for libraries:
/**   SCEERUN2 SCEERUN SCIEH.H SCIEH.SYS.H
/** LNGPRFX is the dataset prefix for library SCCNCMP.
/**
// SET CTGHLQ=CTG.V7R1M0
// SET LTARG=CICSSYSF.APPL65.LOADLIB.PDSE
// SET LIBPRFX=CEE
// SET LNGPRFX=CBC
/**SET CPARMS='LIST,SOURCE,SHOWINC,XPLINK,XREF'
// SET CPARMS='XPLINK'
/**-----
/**      EXEC the C Compiler
/**-----
//COMPILE EXEC PGM=CCNDRVR,
// PARM='/&CPARMS.'
//STEPLIB DD DSNAME=&LIBPRFX..SCEERUN2,DISP=SHR
//        DD DSNAME=&LNGPRFX..SCCNCMP,DISP=SHR
//        DD DSNAME=&LIBPRFX..SCEERUN,DISP=SHR
//SYSLIB  DD DSNAME=&LIBPRFX..SCIEH.H,DISP=SHR
//        DD DSNAME=&LIBPRFX..SCIEH.SYS.H,DISP=SHR
//USERLIB DD DSNAME=&CTGHLQ..SCTGINCL,DISP=SHR
//SYSIN   DD DSN=&CTGHLQ..SCTGSAMP(CTGSMFRD),DISP=SHR
//SYSLIN  DD DSN=&&TEXT,DISP=(,PASS),
//        SPACE=(32000,(80,80)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSPRINT DD SYSOUT=*
//SYSLIST DD SYSOUT=*
//SYSERR  DD SYSOUT=*
/**-----
/**      EXEC the BINDER
/**-----
// IF COMPILE.RC < 5 THEN
//LINK     EXEC PGM=IEWL,

```

```
// PARM='AMODE=31,MAP,DYNAM=DLL,CASE=MIXED,COMPAT=CURR'
//STEPLIB DD DSN=&LIBPRFX..SCEERUN2,DISP=SHR
//          DD DSN=&LIBPRFX..SCEERUN,DISP=SHR
//C8921    DD DSN=&LIBPRFX..SCEELIB,DISP=SHR
//CTGLIB   DD DSN=&CTGHLQ..SCTGSID,DISP=SHR
//SAMPTEXT DD DSN=&&TEXT,DISP=(OLD,DELETE)
//SYSLMOD  DD DSN=&LTARG.(CTGSMFRD),DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSLIB   DD DSN=&LIBPRFX..SCEEBND2,DISP=SHR
//SYSLIN   DD *
INCLUDE SAMPTEXT
INCLUDE C8921(CELHS003)
INCLUDE C8921(CELHS001)
NAME CTGSMFRD(R)
/*
// ENDIF
```

JCL to extract type111 records from SMF

Example C-2 contains the JCL we used to extract type 111 records from the SMF datasets. This job was used in “Accessing the CICS TG SMF records” on page 296.

Example: C-2 JCL used to extract type111 records from SMF

```
//SMFDUMP EXEC PGM=IFASMFDP
//INDD1 DD DSN=SMFDATA.CICSRECS.G4899V00,DISP=SHR
//INDD2 DD DSN=SMFDATA.CICSRECS.G4900V00,DISP=SHR
//INDD3 DD DSN=SMFDATA.CICSRECS.G4901V00,DISP=SHR
//INDD4 DD DSN=SMFDATA.CICSRECS.G4902V00,DISP=SHR
//INDD5 DD DSN=SMFDATA.CICSRECS.G4903V00,DISP=SHR
//OUTDD1 DD DSN=SMFDATA.CTGRECS.REDBOOKS,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
        INDD(INDD1,OPTIONS(DUMP))
        INDD(INDD2,OPTIONS(DUMP))
        INDD(INDD3,OPTIONS(DUMP))
        INDD(INDD4,OPTIONS(DUMP))
        INDD(INDD5,OPTIONS(DUMP))
        OUTDD(OUTDD1,TYPE(111))
/*
```

JCL to run SORT SMF records

Example 8-4 contains the JCL we used to sort the SMF records before CTGSMFRD ran. This job was used in “Accessing the CICS TG SMF records” on page 296.

Important: SMF records are variable block format. The first couple of records are short and caused DFSORT to complete with a RC16 and message:

```
ICE218A 7 18 BYTE VARIABLE RECORD IS SHORTER THAN 57 BYTE MINIMUM  
FOR FIELDS
```

To prevent this error, specify OPTION VLSHRT.

Example 8-4 JCL used to sort records

```
//SORT EXEC PGM=ICEMAN  
//SYSOUT DD SYSOUT=*  
//SORTIN DD DISP=SHR,DSN=SMFDATA.CTGRECS.REDBOOKS  
//SORTOUT DD DISP=SHR,DSN=SMFDATA.CTGRECS.REDBOOKS.SORTED  
//SORTWK01 DD SPACE=(CYL,(10,5)),UNIT=3390  
//SORTWK02 DD SPACE=(CYL,(10,5)),UNIT=3390  
//SORTWK03 DD SPACE=(CYL,(10,5)),UNIT=3390  
//SYSPRINT DD SYSOUT=*  
//SYSIN DD *  
OPTION VLSHRT  
SORT FIELDS=(55,8,CH,A,69,4,BI,A,73,4,BI,A)  
/  

```

JCL to run CICS TG record viewer sample program CTGSMFRD

Example C-3 on page 376 contains the JCL we used to run program CTGSMFRD. This job was used in “Accessing the CICS TG SMF records” on page 296.

This JCL was based on sample job CTGSMFR found in the CICS TG SCTGSAMP.

Example: C-3 JCL used to run program CTGSMFRD

```
//CTGSMFR JOB (0),MSGCLASS=X,CLASS=A,NOTIFY=&SYSUID,REGION=96M
/*
/* Sample JCL to run the SMF record viewer sample program
/* CTGSMFRD.
/*
/* @start_copyright_sample_ZOS@
/* Licensed Materials - Property of IBM
/*
/* 5655-R25
/*
/* (c) Copyright IBM Corp. 2007
/*
/* US Government Users Restricted Rights - Use, duplication or
/* disclosure restricted by GSA ADP Schedule Contract with
/* IBM Corp.
/*
/* Status: Version 7 Release 1
/* @end_copyright_sample_ZOS@
/*
/* Before running this JCL, you will need to modify the JOB
/* control card and globally replace the installation dependent
/* variables:
/*   <Input Dataset>
/* Sequential data set output from the SMF program IFASMFDP.
/* Must contain CICS TG SMF records (Type 111). For information about
/* this program see the z/OS MVS System Management Facilities (SMF).
/*   <EXE PDSE>
/* The user data set containing the compiled and linked CTGSMFRD
/* program. The sample JCL CTGSMFB compiles and links the program.
/*
/* This job will run the CICS TG SMF viewer sample program
/* CTGSMFRD.
/*
/* Definition step: create variables for rest of JCL
/*
/* LTARG the PDSE that contains the CTGSMFRD executable.
/* INARG is the dataset containing the IFASMFDP output.
/*
/*SET LTARG=<Input Dataset>
/*SET INARG=<EXE PDSE>
/* Execute step: run the CICS TG SMF formatter example.
/* With no parameters specified, all CICS TG records in the dataset
/* will be formatted to stdout.
```

```
/* Filtering can be specified by adding options to the PARM parameter
/* on the EXEC statement.
//RUN      EXEC PGM=CTGSMFRD,
//          PARM='-C '
/*          PARM=''
/* The commented example filters for records with an applid CTGAPPL1
/* written on the current day.
//STEPLIB DD DISP=SHR,DSN=CICSSYSF.APPL65.LOADLIB.PDSE
//INPUT   DD DSN=SMFDATA.CTGRECS.REDBOOKS,DISP=SHR
//STDOUT  DD  SYSOUT=*
//STDERR  DD  SYSOUT=*
OPTION PRINT
END
/*
//
```



Additional material

This book refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material associated with this book is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser at:

<ftp://www.redbooks.ibm.com/redbooks/SG247562>

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the IBM Redbooks form number, SG24-7562.

Using the Web material

The additional Web material that accompanies this book includes the following files:

<i>File name</i>	<i>Description</i>
SG247562.zip	Zipped Code Samples

The SG247562.zip file includes the three EAR files that we used in our test scenarios. Table D-1 provides a short description of each EAR file.

Table D-1 Downloadable files

Files	Description
CTGTesterCCIXA.ear	Includes the CTGTesterCCIXA application that we used in our transaction tests in Chapter 7, “High availability with XA and OMEGAMON XE” on page 241.
dfhxcurm.txt	This is a sample EXCI User Replaceable Module used in Chapter 7, “High availability with XA and OMEGAMON XE” on page 241.
cicdelay.txt	A sample CICS assembler program that can delay, for a configurable number of milliseconds, an OPEN TCB.
eciprogr.txt	A sample CICS COBOL application that we used as a DPL back end in our environment.
killcm.sh	A shell script used to cancel the TCP/IP sockets opened by a Gateway daemon. It is used to simulate a network failure in Chapter 7, “High availability with XA and OMEGAMON XE” on page 241.

For further details about the sample J2EE applications, see Appendix A, “Sample J2EE application - CTGTesterCCIXA” on page 353.

System requirements for downloading the Web material

There are no specific system requirements for your workstation to download the supplied materials. However, to be able to use the supplied samples, you need to adhere to product specifications as described throughout this book.

How to use the Web material

Create a subdirectory (folder) on your workstation, and unzip the contents of the Web material zipped file into this folder.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

For information about ordering these publications, see “How to get Redbooks” on page 384. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *Architecting Access to CICS within an SOA*, SG24-5466

Other publications

These publications are also relevant as further information sources:

- ▶ *CICS Application Programming Reference*, SC34-6434
- ▶ *CICS External Interfaces Guide for CICS Transaction Server on z/OS, Version 2 Release 2*, SC34-6449
- ▶ *CICS Transaction Gateway on z/OS Administration*, SC34-6672
- ▶ *CICS Transaction Server on z/OS V3.2 CICS Resource Definition Guide*, SC34-6815
- ▶ *CICS Transaction Server on z/OS V3.2 Internet Guide*, SC34-6831
- ▶ *IBM Tivoli Monitoring Installation and Setup Guide*, GC32-9407
- ▶ *IBM Tivoli Monitoring Services on z/OS Program Directory V6.1.0*, GI11-4105
- ▶ *IBM Tivoli OMEGAMON XE for CICS TG on z/OS: Planning and Configuration Guide*, SC23-5962
- ▶ *IBM Tivoli OMEGAMON XE for CICS TG on z/OS Program Directory V4.1.0*, GI11-8079
- ▶ *Security Server RACF System Programmer's Guide*, SA22-7681
- ▶ *z/OS V1R8.0 Distributed File Service zFS Administration*, SC24-5989
- ▶ *z/OS V1R8.0 UNIX System Services Command Reference*, SA22-7802
- ▶ *z/OS V1R8.0 UNIX System Services Planning*, GA22-7800

Online resources

These Web sites are also relevant as further information sources:

- ▶ CICS TG support pages

<http://www.ibm.com/software/http/cics/ctg/reqs>

- ▶ OMEGAMON XE for CICS TG on z/OS support web site

<http://www-306.ibm.com/software/sysmgmt/products/support/IBMTivoliOMEGAMONXEforCICSTransactionGatewayonZOS.html>

How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

Symbols

/D SMF 163
_BPX_SHAREAS environment variable 41

A

abend code
 S806 156
ABENDBKOUT, EXCI option 363
address space 31
 required sharing 32
AEIO 187
AIX 7
APAR
 PK53163 5
APAR OA21971 101
APAR PK53163 5
APARS
 OA13628 88
 OA22681 88
 OA22683 88
APF authorized 100
APPC connection 22
APPLID 125
ARXC 187

B

BPX.SMF facility 295
BPX_SHAREAS 47
BPXBATCH 63
BPXPRM
 parameters 72

C

C 4
C++ 4
ccf2.jar 30
CCI 14
CEEDUMP 154
changing a password 11
checklist
 software 26
checklist, definitions 28

CICDELAY 150
CICS
 configuration 126
CICS and CICS TG configuration 124
CICS ECI XA resource adapter. 353
CICS EPI resource adapter (cicsepi.rar) 11
CICS EXCI protocol, 19
CICS LU6.2 UOWID 203
CICS PA
 Historical Database (HDB) facility. 5
CICS PA V2.1 5
CICS Password Expiration Management (PEM) 11
CICS resource adapters 16
 cicseci.rar 16
 cicseciXA.rar 16
 cicsepi.rar 16
CICS Server (all) 199
CICS Server (instance) 199
CICS TG
 configuration 33, 124
 configuring 33
 Dynamic Link Library (DLL) 95
 for Linux on System z 22
 for SMF recording 49
 JNI library 23
 monitoring exits 202
 overview 4
 product executables 33
 product file 28
 security tasks 30
 SMF 111 5
 SMF recording 295
 SMP/E installation 28
 statistics categories 200
 STDOUT output 57
 transactional support 39, 46, 56
CICS TG for Multiplatforms 4, 8
CICS TG for RRMS 46
CICS TG on z/OS 4
CICS TG on z/OS 7.1
 Advanced system metrics 5
 Advanced workload monitoring 6
 Channels as modern-day COMMAREAs 6
 Extended Secure Sockets Layer (SSL) support

- 6
- Extended XA support
 - 6
 - Transaction monitoring 5
 - WLM health reporting interval 6
- CICS TG on z/OS V6.1 4
- CICS TG V7.1
 - Migration 69
- CICS TS 6, 9, 26, 87
- CICS Universal Client V6 7
- CICSCLI environment variable 41
- cicseci.rar 10, 16
- cicseciXA.rar 10, 16
- cicsepi.rar 16
- cicsj2ee.jar 30
- CLASSPATH 209
- CLASSPATH environment variable 42
- COBOL 4
- COLUMNS environment variable 43
- COMMAREA 10, 56, 353
- commit optimization 18
- Common Client Interface (CCI) 10–11, 14
- Communications Server for Linux 22
- configuration file 29, 38
 - default name 38
- configuration tool 70
- configuring
 - CICS TG 33
- CONNECTION definition 52
- Connection factory creation 142
- Connection factory customization 144
- connection management contract 14
- Connection Manager 199
 - thread 200
- Connection Manager threads 125
- connection pooling 18–19, 21
- connector.jar 30
- contract
 - connection management 14
 - security 15
 - transaction management 14
- contracts, system-level 14
- creating a Gateway daemon configuration file 38
- creating an HFS data set 34
- creating an STDENV file 40
- creating the started task JCL 45
- credentials 18
- CTG.INI 38, 70
- CTG_MASTER_INPUT 47

- CTG_MASTER_NATLANG 47
- CTG_MASTER_RRMNAME 47
- CTG_MASTER_TRACE_ON 47
- CTG_RRMNAME environment variable 42
- CTG6119E
 - This product will not execute on z/OS.e 162
- CTG6525E
 - Unable to start handler for the tcp protocol 156
- CTG6818E
 - Begin Context failed. 176
- CTG6823E
 - EXCI DPL_REQUEST specific error 175
- CTG6876E
 - EXCI error 170
- CTG6992E
 - No pipe available 169
- CTG6999E
 - Unable to open configuration file 158
- CTG8659E
 - Unable to initialize CTGRRMS services 1003
 - 163
- CTG9300E
 - Writing a record to SMF failed 162
- CTG9638E
 - Transaction Abend 187
- CTG9681E
 - not enabled to support XA 187
- CTGBATCH 207
- ctgclient.jar 30
- CTGDBG 74
- CTGENVVAR 44
- ctgenvvar 29
- ctgenvvarsamp 29
- ctgping 197
- CTGRRMS services 48
- ctgsamp.ini 29
- ctgsamples.jar 30
- ctgserver.jar 30
- CTGSMFRD program 296
- ctgstart 29
- CTGTesterCCIXA application 353–354
- CTGTESTR
 - test output 61
- CTGTESTR sample program 57

D
DASD 106

- data conversion 56
- definitions checklist 28
- DFHCNV 56
- DFHJVPIPE environment variable 43, 167
- DFHJVSYSTEM environment variable 43
- DFHXCURM 261
- DFHXCURM user exit 263
- DFSORT 297
- distributed program link (DPL) 10
- dynamic trace 79, 82
- dynamic trace (JNI) 82
- Dynamic Workspace Links 129

E

- EBCDIC 56
- EC01, COBOL example 56, 61
- ECI
 - error codes 153
 - password, verify 11
- ECI request 10, 58, 200
- ECI_ERR_NO_CICS 62, 80, 263
- ECI_ERR_NO_CICS -3 166
- ECI_ERR_RESOURCE_SHORTAGE 263
- ECI_ERR_RESOURCE_SHORTAGE -16 169
- ECI_ERR_SECURITY_ERROR -27 177
- ECI_ERR_SYSTEM_ERROR 263
- ECI_ERR_SYSTEM_ERROR -9 175
- ECIPROG 56, 150
- EDC5111I
 - Permission denied 158
- EDC8115I
 - Address already in use 157
- Enterprise Information System (EIS) 13
- environment variable 40, 45, 74
 - _BPX_SHAREAS 41
 - CICSCLI 41
 - CLASSPATH 42
 - COLUMNS 43
 - CTG_RRMNAME 42
 - DFHJVPIPE 43, 167
 - DFHJVSYSTEM 43
 - PATH 43
 - search order 44
 - STEPLIB 43
 - TMPDIR 43
 - TZ 44
- EPI 10
- EPI support classes 11

- EPIRequest 10
- ERROR_CONNECTION_FAILED 188
- ERROR_WORK_WAS_REFUSED61445(0xF005) 178
- ESI 11
 - password, change 11
- EWLM 203
- EXCI 20
- EXCI connection 52
- EXCI interface 5
- EXCI NETNAME 126
- EXCI pipe 53, 261
- EXCI requests 198
- EXCI sync-on-return 203
- EXCI trace 83
- EXCI_LOADLIB 156
- External Call Interface 10
- External Call Interface. *See* ECI.
- External Presentation Interface. *See* EPI.
- External Security Interface. *See* ESI.
- external security manager (ESM) 11

F

- FACILITY class, RACF 31
- files
 - ccf2.jar 30
 - cicseci.rar 10, 16
 - cicseciXA.rar 10, 16
 - cicsepi.rar 16
 - cicsj2ee 30
 - connector.jar 30
 - ctgclient.jar 30
 - ctgsamp.ini 29
 - ctgsamples.jar 30
 - ctgserver.jar 30
 - guide.jar 30
 - psk.jar 30
 - STDENV 40

G

- Gateway Daemon 199
 - Statistics Collection 99
- Gateway daemon 7, 19, 37, 154, 353, 355
 - dummy DD statement 74
 - file system I/O constraint 222
 - RACF permissions 65
 - Worker thread constraint 213
- Gateway daemon configuration file 38

- Gateway daemon statistics 198
- Gateway trace 79
- Generalized Trace Facility (GTF) 78
- Generation Dataset Group (GDG) 301
- get-use-cache model 15
- global transaction 353
- guide.jar 30

H

- Health interval (seconds) 125
- Health reporting 125
- health reporting 262
- healthinterval 268
- HFS
 - Directory layout 33
 - directory permissions 37
- HFS data set, creating 34
- HFS data set, mounting 35
- HFS directory 42
- High availability 284
- HiperSockets 22
- Host name 125
- HP-UX 7

I

- IBM SDK 26, 87
- IBM SDK on z/OS, Java 2 Technology Edition, V1.5 27
- IBM Tivoli Monitoring
 - components 88
- IBM z/VM® 22
- IEFBR14 batch job 34
- IOException CTG6651E 186
- IOException CTG6668E 188
- IP address 125
- IPCS
 - utility 83
- IPIC sessions 21
- ITMS
 - Engine V6.1.0 87

J

- J2C Connection Factory 243
- J2EE 4
- J2EE application server 4
- J2EE Connector Architecture. *See* JCA.
- Java 3

- Java 2 Platform Enterprise Edition 10
- Java 5 23
- Java Development Kit 72
- Java exceptions 153
- JCA 13
 - authentication entry 18
 - overview 13
 - Version 1.5 15
 - with different topologies 16
- JCL
 - CTGTESTR 58
 - for Gateway daemon started task 46
 - for master process SCSC710 47
 - used to extract type111 records from SMF 374
 - used to link-edit CTGSMFRD 372
 - used to run program CTGSMFRD 376
 - used to sort records 375
- JDBC 14
- JES 154
- JNI interface 22
- JNI trace 78, 81
- Job name 125
- JVM
 - system properties 207
- JVM dumps 154

L

- Language Environment 76
- last-participant support 18
- last-resource commit optimization 18
- lazy connection association 15
- lazy transaction enlistment 15
- LEOPTS 76
- Linux 7
- Linux on zSeries 22
- local connection 353
- local mode of operation 12
- local transaction 362
- LPAR 129

M

- mounting an HFS data set 35

N

- NETNAME 125
- NETNAME, CONNECTION definition 28, 65
- netstat, TCP/IP test tool 71

Network delay 211
Network failure 278

O

OMEGAMON XE
 components installed on z/OS 127
 configuration 127
OMEGAMON XE environment 94
OMEGAMON XE for CICS TG
 historical function 294
OMEGAMON XE for CICS TG on z/OS 85, 196
OMEGAMON XE for CICS TG on z/OS V4.1.0 87
OMVS 72
OMVS segment 30
overview of CICS TG 4

P

password, changing 11
password, verify 11
PATH environment variable 43
PDSE library 296
Problem scenarios 210
PROCLIB 100
Protocol handler 199
psk.jar 30

Q

QR_TCB 230

R

RACF 11
 BPX.FILEATTR.PROGCTL FACILITY class 31
 BPX.SERVER FACILITY class 31
 exit 31
 profile 30
RACF security tasks 30
Rational Application Developer (RAD) 356, 363
RECEIVECOUNT 53
Redbooks Web site 384
 Contact us xiii
remote mode of operation 12
remote-mode 353
resource adapter 14–16, 23
Resource adapters 14
resource manager (RM) 56
Resource Recovery Management Service (RRMS)
39, 56

Resource Recovery Service (RRS) 42
RRMS, SIT parameter 56

S

S806
 steplib incorrect 155–156
sample application 353
SCEERUN2 32
scripts
 ctgenvvarsamp 29
 ctgstart 29
SCTGLOAD 32
SDFHEXCI 32
SDFHLINK 32
SDK 23
SDSF 224
SDSF log 63
SDSF PS command 78
security
 considerations 30
 tasks 30
security contract 15
security credentials 18
security management 18, 20–21
SESSIONS definition 52–53, 67
setting permissions 37
SMF
 Interval statistics and off-line recording 5
SMF parms 51
SMF statistics 305
SMF111 39
SMP/E installation 28
SMP/E target libraries 106
SNA network connections 18
SnapTrace 154
Software checklist 86
software checklist 26
Software components 86
SSL
 protocol 7
static trace 79, 81
Static trace (JNI) 81
Stats DLL dataset 125
Stats end of day 125
Stats port 125
STDENV 40
STDENV DD
 card 44

- statement 41, 48
- STDENV file 40
- STDERR DD 154
- STDOUT 154
 - Gateway daemon start 60
- STDOUT. 60
- STEPLIB environment variable 43
- storm drain 261
- SupportPac CH50 197
- SVC dump 154
- SYSPLEXWLMPOLL 262
- System Environment 199
- System Monitoring Facility (SMF) 293
- system-level contracts 14
- Systems Network Architecture (SNA) 17
- SystemSSL 70

T

- TCP
 - protocol 7
- TCP port 125
- TCP/IP 17
 - port 122
- TCP/IP port 28
- TEP
 - Browser client 140
 - Desktop client 140
- Tivoli Data Warehouse (TDW) 91
- Tivoli Enterprise Monitoring Agents (TEMA) 90
- Tivoli Enterprise Monitoring Server (TEMS) 89
- Tivoli Enterprise Portal (TEP) client 90
- Tivoli Enterprise Portal Server (TEPS) 90
- TMPDIR environment variable 43
- topologies 16
- tracing 78
 - creating a data set 33
 - dynamic trace 79, 82
 - EXCI trace 78, 83
 - Gateway trace 79
 - JNI trace 78, 81
 - static trace 79, 81
- transaction attribute 361
- transaction management 18, 20–21
- transaction management contract 14
- transaction manager 362
- two-phase commit 18, 21
- TZ environment variable 44

U

- UID 31
- Unable to allocate bytes for GC in
 - j9vmem_reserve_memory 159
- UNICODE 56
- UNIX System Services (USS) 155
- UNIX System Services program control 31
- user ID 30

V

- verifying a password 11
- virtual private networks 20
- VPN 20

W

- WebSphere Application Server 4, 13, 15, 17, 19
- WebSphere Application Server on z/OS 20
- Windows 2000 7
- Windows 2003 7
- Windows XP 7
- Worker thread 199
- Worker threads 125
- WTRN0078E 188

X

- XA 1PC configuration 243
- XA 2PC configuration 284
- XA support 125
- XA transaction 6, 9, 354
- XA transactions 203
- XA two-phase commit (2PC) 242

Z

- z/OS Configuration Tool 95
- z/OS TCP/IP commands 71
- z/OS UNIX System Services 26
- z/OS V1R8 87
- zSeries File System (zFS) 35



Exploring Systems Monitoring for CICS Transaction Gateway V7.1 on z/OS

(0.5" spine)
0.475" <-> 0.875"
250 <-> 459 pages



Exploring Systems Monitoring for CICS Transaction Gateway V7.1

**Discover the power
of Tivoli OMEGAMON
XE for CICS TG on
z/OS**

**Explore real life high
availability scenarios**

**Understand the value
of SMF historical
data**

How can you manage your CICS Transaction Gateway systems on z/OS so as to meet the growing availability and performance requirements of your business?

The CICS Transaction Gateway V7.1 provides a wealth of new systems monitoring functionality, and together with the new IBM Tivoli OMEGAMON XE for CICS TG product, provides you with a set of enhanced capacity planning and problem determination capabilities.

The first part of this IBM Redbooks publication concentrates on product installation and customization for both the CICS Transaction Gateway V7.1 on z/OS and IBM Tivoli OMEGAMON XE for CICS TG product. The second part of the book highlights the new systems monitoring functionality in CICS Transaction Gateway V7.1 on z/OS and how it is complemented by the new IBM Tivoli OMEGAMON XE for CICS TG product. A set of typical customer scenarios are used to demonstrate the practical usage of the new functions, in the following four problem determination areas:

- ▶ Diagnosing common problems
- ▶ Diagnosing system slow downs
- ▶ High Availability with XA and OMEGAMON XE
- ▶ Historical data analysis

Together, these scenarios can be used to quickly determine the root cause of typical production problems and provide in-depth information to allow you to plan, build, and monitor a highly available CICS Transaction Gateway systems to provide SOA access to your existing CICS applications.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks